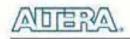
# DEO-Nano User Manual World Leading FPGA Based Products and Design Services





Copyright @ 2003-2013 Terasic Technologies Inc. All Rights Reserved.



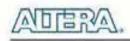


CHAPTER 1	INTRODUCTION	5
1.1 Features		5
1.2 About the KIT	,	7
1.3 Getting Help		7
CHAPTER 2	DE0-NANO BOARD ARCHITECTURE	8
2.1 Layout and Co	omponents	
2.2 Block Diagram	n of the DE0-Nano Board	9
2.3 Power-up the I	DE0-Nano Board	
CHAPTER 3	USING THE DE0-NANO BOARD	
3.1 Configuring th	e Cyclone IV FPGA	
3.2 General User I	Input/Output	
3.3 SDRAM Mem	10 <b>r</b> y	
3.4 I2C Serial EEF	PROM	
3.5 Expansion Hea	aders	
3.6 A/D Converter	and 2x13 Header	
3.7 Digital Acceler	rometer	
3.8 Clock Circuitry	у	
3.9 Power Supply		
CHAPTER 4	DE0-NANO CONTROL PANEL	
4.1 Control Panel	Setup	
4.2 Controlling the	e LEDs	
4.3 Switches and F	Pushbuttons	
4.4 Memory Contr	roller	
4.5 Digital Acceler	rometer	
4.6 ADC		
4.7 Overall Structu	ure of the DE0-Nano Control Panel	
CHAPTER 5	DE0-NANO SYSTEM BUILDER	
5.1 Introduction		

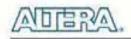
2

# ADERA.

5.2 General Design	Flow	
5.3 Using DE0-Nand	o System Builder	
CHAPTER 6	TUTORIAL: CREATING AN FPGA PROJECT	
6.1 Design Flow		
6.2 Before You Begi	n	
6.3 What You Will I	earn	
6.4 Assign The Devi	ice	
6.5 Creating an FPG	A design	
6.6 Assign the Pins.		
6.7 Create a Default	TimeQuest SDC File	
6.8 Compile Your D	esign	
6.9 Program the FPO	GA Device	
6.10 Verify The Har	dware	
CHAPTER 7	TUTORIAL: CREATING A NIOS II PROJECT	
7.1 Required Feature	es	
7.2 Creation of Hard	lware Design	
7.3 Download the H	ardware Design	
7.4 Create a hello_w	vorld Example Project	
7.5 Build and Run th	ne Program	
7.6 Edit and Re-Rur	the Program	
7.7 Why the LED B	links	
7.8 Debugging the A	Application	
7.9 Configure System	m Library	
CHAPTER 8	DE0-NANO DEMONSTRATIONS	
8.1 System Requirer	nents	
8.2 Breathing LEDs		
8.3 ADC Reading		
8.4 SOPC Demo		
8.5 G-Sensor		
8.6 SDRAM Test by	v Nios II	
CHAPTER 9	APPENDIX	
9.1 Programming th	e Serial Configuration Device	
9.2 EPCS Programn	ning via nios-2-flash-programmer	
terasic Terasic D	BEO-Nano User Manual	www.terasic.com



9.3 Revision History	155
9.4 Copyright Statement	155



## Chapter 1



The DE0-Nano board introduces a compact-sized FPGA development platform suited for to a wide range of portable design projects, such as robots and mobile projects.

The DE0-Nano is ideal for use with embedded soft processors—it features a powerful Altera Cyclone IV FPGA (with 22,320 logic elements), 32 MB of SDRAM, 2 Kb EEPROM, and a 64 Mb serial configuration memory device. For connecting to real-world sensors the DE0-Nano includes a National Semiconductor 8-channel 12-bit A/D converter, and it also features an Analog Devices 13-bit, 3-axis accelerometer device.

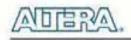
The DE0-Nano board includes a built-in USB Blaster for FPGA programming, and the board can be powered either from this USB port or by an external power source. The board includes expansion headers that can be used to attach various Terasic daughter cards or other devices, such as motors and actuators. Inputs and outputs include 2 pushbuttons, 8 user LEDs and a set of 4 dip-switches.

## **1.1 Features**

Figure 1-1 shows a photograph of the DE0-Nano Board.

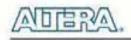






The key features of the board are listed below:

- Featured device
  - Altera Cyclone® IV EP4CE22F17C6N FPGA
  - o 153 maximum FPGA I/O pins
- Configuration status and set-up elements
  - On-board USB-Blaster circuit for programming
  - Spansion EPCS64
- Expansion header
  - Two 40-pin Headers (GPIOs) provide 72 I/O pins, 5V power pins, two 3.3V power pins and four ground pins
- Memory devices
  - 32MB SDRAM
  - o 2Kb I2C EEPROM
- General user input/output
  - o 8 green LEDs
  - o 2 debounced pushbuttons
  - o 4-position DIP switch
- G-Sensor
  - ADI ADXL345, 3-axis accelerometer with high resolution (13-bit)
- A/D Converter
  - NS ADC128S022, 8-Channel, 12-bit A/D Converter
  - o 50 Ksps to 200 Ksps
- Clock system
  - On-board 50MHz clock oscillator
- Power Supply
  - USB Type mini-AB port (5V)
  - DC 5V pin for each GPIO header (2 DC 5V pins)
  - 2-pin external power header (3.6-5.7V)



## **1.2 About the KIT**

The kit comes with the following contents:

- DE0-Nano board
- System CD-ROM.
- USB Cable

The system CD contains technical documents for the DE0-Nano board, which includes component datasheets, demonstrations, schematic, and user manual.

Figure 1-2 shows the photograph of the DE0-Nano kit contents.

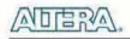


Figure 1-2 DE0-Nano kit package contents

## **1.3 Getting Help**

Here is information of how to get help if you encounter any problem:

- Terasic Technologies
- Tel: +886-3-575-0880
- Email: support@terasic.com
- Altera Corporation
- Email: university@altera.com



## Chapter 2

# **DE0-Nano Board Architecture**

This chapter describes the architecture of the DE0-Nano board including block diagram and components.

## **2.1 Layout and Components**

The picture of the DE0-Nano board is shown in **Figure 2-1** and **Figure 2-2**. It depicts the layout of the board and indicates the locations of the connectors and key components.

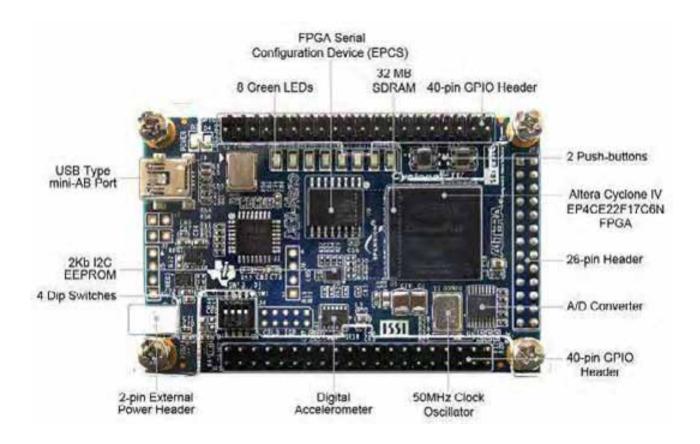


Figure 2-1 The DE0-Nano Board PCB and component diagram (top view)

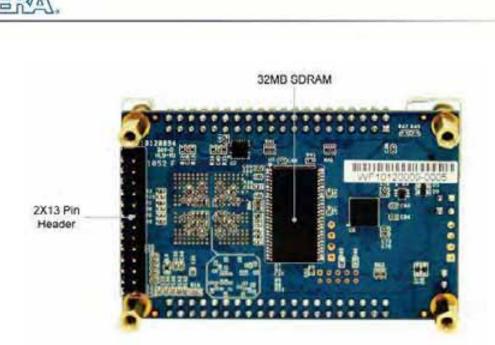


Figure 2-2 The DE0-Nano Board PCB and component diagram (bottom view)

## **2.2 Block Diagram of the DE0-Nano Board**

**Figure 2-3** shows the block diagram of the DE0-Nano board. To provide maximum flexibility for the user, all connections are made through the Cyclone IV FPGA device. Thus, the user can configure the FPGA to implement any system design.

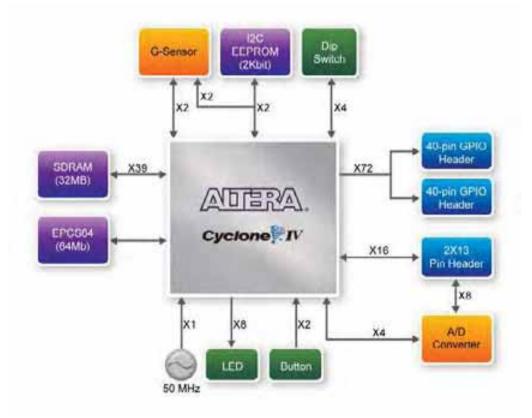


Figure 2-3 Block diagram of DE0-Nano Board

## **2.3 Power-up the DE0-Nano Board**

The DE0-Nano board comes with a preloaded configuration bit stream to demonstrate some features of the board. This allows users to see quickly if the board is working properly. To power-up the board two options are available which are described below:

1. Connect a USB Mini-B cable between a USB (Type A) host port and the board. For communication between the host and the DEO-Nano board, it is necessary to install the Altera USB Blaster driver software.

2. Alternatively, users can power-up the DE0-Nano board by supplying 5V to the two DC +5 (VCC5) pins of the GPIO headers or supplying (3.6-5.7V) to the 2-pin header.

At this point you should observe flashing LEDs on the board.



# Using the DE0-Nano Board

This chapter gives instructions for using the DEO-Nano board and describes in detail its components and connectors, along with the required pin assignments.

## **3.1 Configuring the Cyclone IV FPGA**

The DE0-Nano board contains a Cyclone IV E FPGA which can be programmed using JTAG programming. This allows users to configure the FPGA with a specified design using Quartus II software. The programmed design will remain functional on the FPGA as long as the board is powered on, or until the device is reprogrammed. The configuration information will be lost when the power is turned off.

To download a configuration bit stream file using JTAG Programming into the Cyclone IV FPGA, perform the following steps:

1. Connect a USB Mini-B cable between a host computer and the DE0-Nano.

2. The FPGA can now be programmed through the Quartus II Programmer by selecting a configuration bit stream file with the .sof filename extension.

#### ■ Configuring the Spansion EPCS64 device

The DE0-Nano board contains a Spansion EPCS64 serial configuration device. This device provides non-volatile storage of the configuration bit-stream, so that the information is retained even when the power supply to the DE0-Nano board is turned off. When the board's power is turned on, the configuration data in the EPCS64 device is automatically loaded into the Cyclone IV E FPGA.

The Cyclone IV E device supports in-system programming of a serial configuration device using the JTAG interface via the serial flash loader design. The serial flash loader is a bridge design for the Cyclone IV E device that uses its JTAG interface to access the EPCS .jic file and then uses the AS interface to program the EPCS device. **Figure 3-1** illustrates the programming method when adopting a serial flash loader solution. Chapter 9 of this document describes how to load a circuit to the serial configuration device.

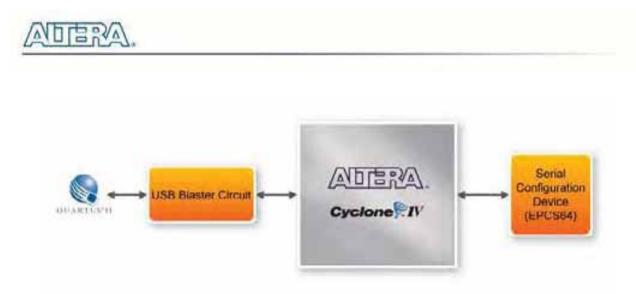
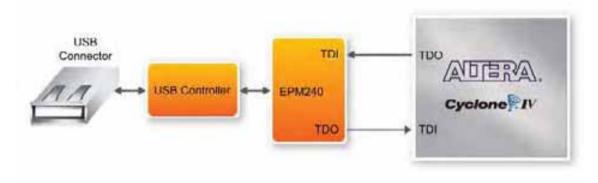


Figure 3-1 Programming a serial configuration device with serial flash loader solution

#### ■ JTAG Chain on DE0-Nano Board

The JTAG Chain on the DE0-Nano board is connected to a host computer using an on-board USB-blaster. The USB-blaster consists of a USB Mini-B connector, a FTDI USB 2.0 Controller, and an Altera MAX II CPLD.

Figure 3-2 illustrates the JTAG configuration setup.



#### Figure 3-2 JTAG Chain

### **3.2 General User Input/Output**

#### Pushbuttons

The DE0-Nano board contains two pushbuttons shown in **Figure 3-3**. Each pushbutton is debounced using a Schmitt Trigger circuit, as indicated in **Figure 3-4**. The two outputs called KEY0, and KEY1 of the Schmitt Trigger devices are connected directly to the Cyclone IV E FPGA. Each pushbutton provides a high logic level when it is not pressed, and provides a low logic level when pressed. Since the pushbuttons are debounced, they are appropriate for using as clock or reset inputs.

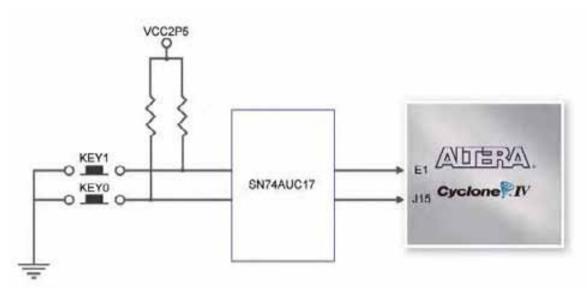


Figure 3-3 Connections between the push-buttons and Cyclone IV FPGA

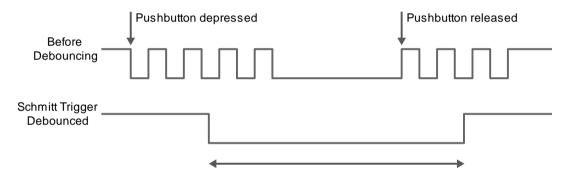
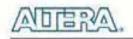


Figure 3-4 Pushbuttons debouncing

#### ■ LEDs

There are 8 green user-controllable LEDs on the DE0-Nano board. The eight LEDs, which are presented in **Figure 3-4**, allow users to display status and debugging information. Each LED is driven directly by the Cyclone IV E FPGA. Each LED is driven directly by a pin on the Cyclone IV E FPGA; driving its associated pin to a high logic level turns the LED on, and driving the pin low turns it off.



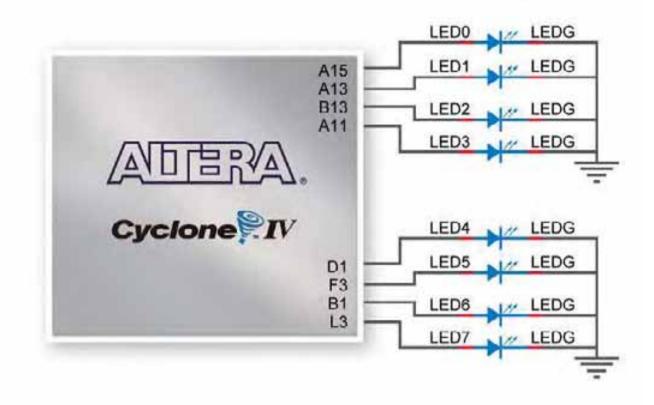


Figure 3-5 Connections between the LEDs and Cyclone IV FPGA

#### ■ **DIP** Switch

The DE0-Nano board contains a 4 dip switches. A DIP switch provides, to the FPGA, a high logic level when it is in the DOWN position, and a low logic level when in the UPPER position.

Table 3-1     Pin Assignments for Push-buttons				
Signal Name	FPGA Pin No.	Description	I/O Standard	
KEY[0]	PIN_J15	Push-button[0]	3.3V	
KEY[1]	PIN_E1	Push-button[1]	3.3V	

Signal Name	FPGA Pin No.	Description	I/O Standard
LED[0]	PIN_A15	LED Green[0]	3.3V
LED[1]	PIN_A13	LED Green[1]	3.3V
LED[2]	PIN_B13	LED Green[2]	3.3V
LED[3]	PIN_A11	LED Green[3]	3.3V
LED[4]	PIN_D1	LED Green[4]	3.3V
LED[5]	PIN_F3	LED Green[5]	3.3V
LED[6]	PIN_B1	LED Green[6]	3.3V
LED[7]	PIN_L3	LED Green[7]	3.3V

#### Table 3-2 Pin Assignments for LEDs



Signal Name	FPGA Pin No.	Description	I/O Standard
DIP Switch[0]	PIN_M1	DIP Switch[0]	3.3V
DIP Switch[1]	PIN_T8	DIP Switch[1]	3.3V
DIP Switch[2]	PIN_B9	DIP Switch[2]	3.3V
DIP Switch[3]	PIN_M15	DIP Switch[3]	3.3V

## **3.3 SDRAM Memory**

The board features a Synchronous Dynamic Random Access Memory (SDRAM) device providing 32MB with a 16-bit data lines connected to the FPGA. The chip uses 3.3V LVCMOS signaling standard. All signals are registered on the positive edge of the clock signal, DRAM\_CLK. Connections between the FPGA and SDRAM chips are shown in **Figure 3-6**.



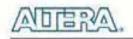
Figure 3-6 Connections between FPGA and SDRAM

Signal Name	FPGA Pin No.	Description	I/O Standard
DRAM_ADDR[0]	PIN_P2	SDRAM Address[0]	3.3V
DRAM_ADDR[1]	PIN_N5	SDRAM Address[1]	3.3V
DRAM_ADDR[2]	PIN_N6	SDRAM Address[2]	3.3V
DRAM_ADDR[3]	PIN_M8	SDRAM Address[3]	3.3V
DRAM_ADDR[4]	PIN_P8	SDRAM Address[4]	3.3V
DRAM_ADDR[5]	PIN_T7	SDRAM Address[5]	3.3V
DRAM_ADDR[6]	PIN_N8	SDRAM Address[6]	3.3V
DRAM_ADDR[7]	PIN_T6	SDRAM Address[7]	3.3V
DRAM_ADDR[8]	PIN_R1	SDRAM Address[8]	3.3V
DRAM_ADDR[9]	PIN_P1	SDRAM Address[9]	3.3V
DRAM_ADDR[10]	PIN_N2	SDRAM Address[10]	3.3V
DRAM_ADDR[11]	PIN_N1	SDRAM Address[11]	3.3V



## 3.4 I2C Serial EEPROM

The DE0-Nano contains a 2Kbit Electrically Erasable PROM (EEPROM). The EEPROM is configured through a 2-wire I2C serial interface. The device is organized as one block of 256 x 8-bit memory. The I2C write and read address are 0xA0 and 0xA1, respectively. **Figure 3-7** illustrates its connections with the Cyclone IV FPGA.



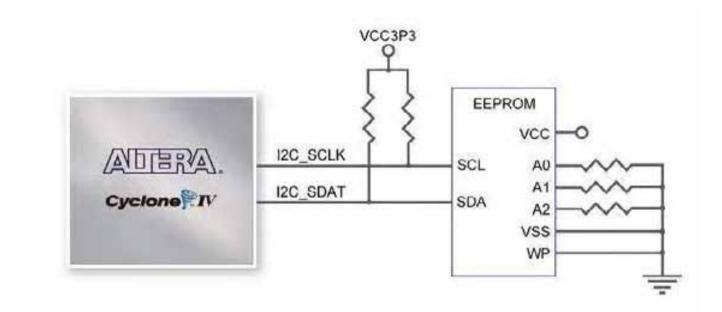


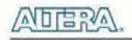
Figure 3-7 Connections between FPGA and EEPROM

Signal Name	FPGA Pin No.	Description	I/O Standard
I2C_SCLK	PIN_F2	EEPROM clock	3.3V
I2C_SDAT	PIN_F1	EEPROM data	3.3V

Table 3-5 Pin	Assignments for	I2C Serial EEPROM

## **3.5 Expansion Headers**

The DE0-Nano board provides two 40-pin expansion headers. Each header connects directly to 36 pins of the Cyclone IV E FPGA, and also provides DC +5V (VCC5), DC +3.3V (VCC33), and two GND pins. **Figure 3-8** shows the I/O distribution of the GPIO connectors.



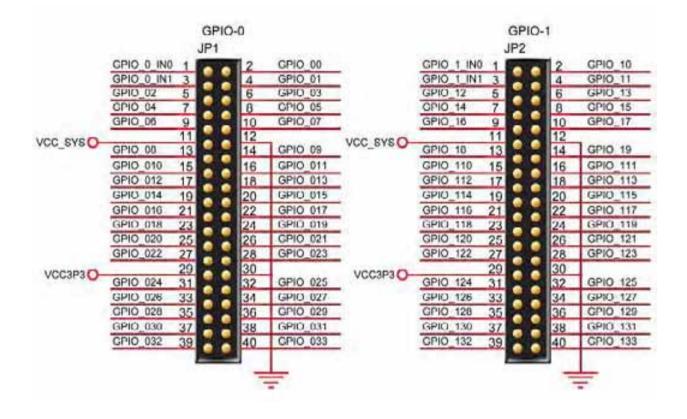


Figure 3-8 Pin arrangement of the GPIO expansion headers

The pictures below indicate the pin 1 location of the expansion headers.

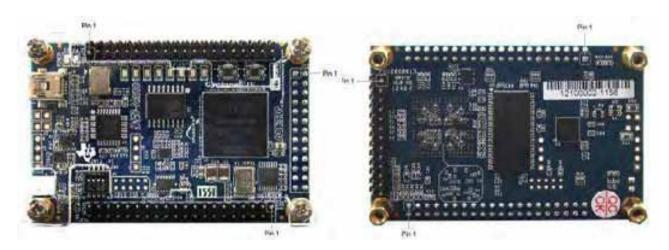


Figure 3-9 Pin1 locations of the GPIO expansion headers

	8			
Signal Name	FPGA Pin No.	Description	I/O Standard	
GPIO_0_IN0	PIN_A8	<b>GPIO Connection DATA</b>	3.3V	
GPIO_00	PIN_D3	GPIO Connection DATA	3.3V	
GPIO_0_IN1	PIN_B8	GPIO Connection DATA	3.3V	
GPIO_01	PIN_C3	GPIO Connection DATA	3.3V	

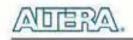
#### Table 3-6 GPIO-0 Pin Assignments



GPIO_02	PIN_A2	<b>GPIO Connection DATA</b>	3.3V
GPIO_03	PIN_A3	<b>GPIO Connection DATA</b>	3.3V
GPIO_04	PIN_B3	<b>GPIO Connection DATA</b>	3.3V
GPIO_05	PIN_B4	<b>GPIO Connection DATA</b>	3.3V
GPIO_06	PIN_A4	GPIO Connection DATA	3.3V
GPIO_07	PIN_B5	<b>GPIO Connection DATA</b>	3.3V
GPIO_08	PIN_A5	GPIO Connection DATA	3.3V
GPIO_09	PIN_D5	GPIO Connection DATA	3.3V
GPIO_010	PIN_B6	GPIO Connection DATA	3.3V
GPIO_011	PIN_A6	GPIO Connection DATA	3.3V
GPIO_012	PIN_B7	GPIO Connection DATA	3.3V
GPIO_013	PIN_D6	GPIO Connection DATA	3.3V
GPIO_014	PIN_A7	GPIO Connection DATA	3.3V
GPIO_015	PIN_C6	GPIO Connection DATA	3.3V
GPIO_016	PIN_C8	GPIO Connection DATA	3.3V
GPIO_017	PIN_E6	GPIO Connection DATA	3.3V
GPIO_018	PIN_E7	GPIO Connection DATA	3.3V
GPIO_019	PIN_D8	<b>GPIO Connection DATA</b>	3.3V
GPIO_020	PIN_E8	GPIO Connection DATA	3.3V
GPIO_021	PIN_F8	GPIO Connection DATA	3.3V
GPIO_022	PIN_F9	GPIO Connection DATA	3.3V
GPIO_023	PIN_E9	<b>GPIO Connection DATA</b>	3.3V
GPIO_024	PIN_C9	GPIO Connection DATA	3.3V
GPIO_025	PIN_D9	GPIO Connection DATA	3.3V
GPIO_026	PIN_E11	GPIO Connection DATA	3.3V
GPIO_027	PIN_E10	GPIO Connection DATA	3.3V
GPIO_028	PIN_C11	GPIO Connection DATA	3.3V
GPIO_029	PIN_B11	GPIO Connection DATA	3.3V
GPIO_030	PIN_A12	GPIO Connection DATA	3.3V
GPIO_031	PIN_D11	GPIO Connection DATA	3.3V
GPIO_032	PIN_D12	GPIO Connection DATA	3.3V
GPIO_033	PIN_B12	GPIO Connection DATA	3.3V

#### Table 3-7 GPIO-1 Pin Assignments

Signal Name	FPGA Pin No.	Description	I/O Standard
GPIO_1_IN0	PIN_T9	<b>GPIO Connection DATA</b>	3.3V
GPIO_10	PIN_F13	GPIO Connection DATA	3.3V
GPIO_1_IN1	PIN_R9	GPIO Connection DATA	3.3V
GPIO_11	PIN_T15	GPIO Connection DATA	3.3V
GPIO_12	PIN_T14	GPIO Connection DATA	3.3V
GPIO_13	PIN_T13	GPIO Connection DATA	3.3V
GPIO_14	PIN_R13	GPIO Connection DATA	3.3V
GPIO_15	PIN_T12	GPIO Connection DATA	3.3V



GPIO_16	PIN_R12	<b>GPIO Connection DATA</b>	3.3V
GPIO_17	PIN_T11	<b>GPIO Connection DATA</b>	3.3V
GPIO_18	PIN_T10	GPIO Connection DATA	3.3V
GPIO_19	PIN_R11	<b>GPIO Connection DATA</b>	3.3V
GPIO_110	PIN_P11	GPIO Connection DATA	3.3V
GPIO_111	PIN_R10	<b>GPIO Connection DATA</b>	3.3V
GPIO_112	PIN_N12	GPIO Connection DATA	3.3V
GPIO_113	PIN_P9	<b>GPIO Connection DATA</b>	3.3V
GPIO_114	PIN_N9	<b>GPIO Connection DATA</b>	3.3V
GPIO_115	PIN_N11	<b>GPIO Connection DATA</b>	3.3V
GPIO_116	PIN_L16	<b>GPIO Connection DATA</b>	3.3V
GPIO_117	PIN_K16	<b>GPIO Connection DATA</b>	3.3V
GPIO_118	PIN_R16	<b>GPIO Connection DATA</b>	3.3V
GPIO_119	PIN_L15	<b>GPIO Connection DATA</b>	3.3V
GPIO_120	PIN_P15	<b>GPIO Connection DATA</b>	3.3V
GPIO_121	PIN_P16	<b>GPIO Connection DATA</b>	3.3V
GPIO_122	PIN_R14	<b>GPIO Connection DATA</b>	3.3V
GPIO_123	PIN_N16	<b>GPIO Connection DATA</b>	3.3V
GPIO_124	PIN_N15	<b>GPIO Connection DATA</b>	3.3V
GPIO_125	PIN_P14	<b>GPIO Connection DATA</b>	3.3V
GPIO_126	PIN_L14	<b>GPIO Connection DATA</b>	3.3V
GPIO_127	PIN_N14	<b>GPIO Connection DATA</b>	3.3V
GPIO_128	PIN_M10	<b>GPIO Connection DATA</b>	3.3V
GPIO_129	PIN_L13	GPIO Connection DATA	3.3V
GPIO_130	PIN_J16	GPIO Connection DATA	3.3V
GPIO_131	PIN_K15	<b>GPIO Connection DATA</b>	3.3V
GPIO_132	PIN_J13	<b>GPIO Connection DATA</b>	3.3V
GPIO_133	PIN_J14	<b>GPIO Connection DATA</b>	3.3V

### **3.6 A/D Converter and 2x13 Header**

The DE0-Nano contains an ADC128S022 lower power, eight-channel CMOS 12-bit analog-to-digital converter. This A-to-D provides conversion throughput rates of 50 ksps to 200 ksps. It can be configured to accept up to eight input signals at inputs IN0 through IN7. This eight input signals are connected to the 2x13 header, as shown in Figure 3-10. The remaining I/Os of the 2x13 header are a DC +3.3V (VCC33), a GND and 13 pins, which are connect directly to the Cyclone IV E device.

For more detailed information on the A/D converter chip, please refer to its datasheet which is available on manufacturer's website or under the /datasheet folder of the system CD.



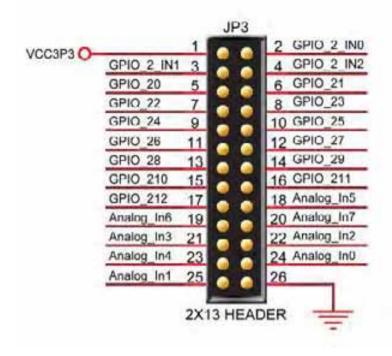


Figure 3-10 Pin distribution of the 2x13 Header

Figure 3-11 shows the connections on the 2x13 header, A/D converter and Cyclone IV device.

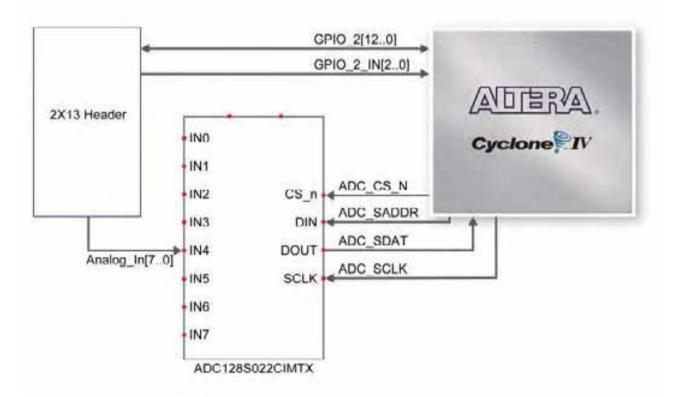


Figure 3-11 Wiring for 2x13 header and A/D converter

The pictures below indicate the pin 1 location of the 2x13 header.



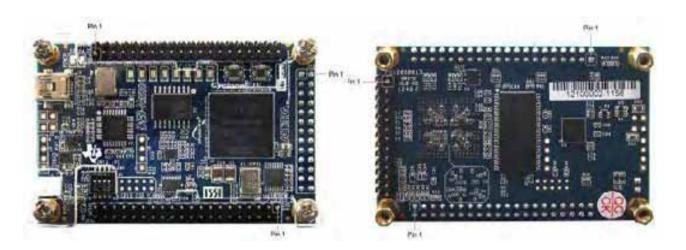


Figure 3-12 Pin1 locations of the 2x13 header

Table 3-8 Pin Assignments for 2x13 Header	Fable 3-8	<b>Pin Assignments</b>	for 2x13	Header
---	-----------	------------------------	----------	--------

Signal Name	FPGA Pin No.	Description	I/O Standard
GPIO_2[0]	PIN_A14	<b>GPIO Connection DATA[0]</b>	3.3V
GPIO_2[1]	PIN_B16	<b>GPIO Connection DATA[1]</b>	3.3V
GPIO_2[2]	PIN_C14	<b>GPIO Connection DATA[2]</b>	3.3V
GPIO_2[3]	PIN_C16	<b>GPIO Connection DATA[3]</b>	3.3V
GPIO_2[4]	PIN_C15	<b>GPIO Connection DATA[4]</b>	3.3V
GPIO_2[5]	PIN_D16	<b>GPIO Connection DATA[5]</b>	3.3V
GPIO_2[6]	PIN_D15	<b>GPIO Connection DATA[6]</b>	3.3V
GPIO_2[7]	PIN_D14	GPIO Connection DATA[7]	3.3V
GPIO_2[8]	PIN_F15	GPIO Connection DATA[8]	3.3V
GPIO_2[9]	PIN_F16	<b>GPIO Connection DATA[9]</b>	3.3V
GPIO_2[10]	PIN_F14	<b>GPIO Connection DATA[10]</b>	3.3V
GPIO_2[11]	PIN_G16	<b>GPIO Connection DATA[11]</b>	3.3V
GPIO_2[12]	PIN_G15	GPIO Connection DATA[12]	3.3V
GPIO_2_IN[0]	PIN_E15	GPIO Input	3.3V
GPIO_2_IN[1]	PIN_E16	GPIO Input	3.3V
GPIO_2_IN[2]	PIN_M16	GPIO Input	3.3V

 Table 3-9 Pin Assignments for ADC

Signal Name	FPGA Pin No.	Description	I/O Standard
ADC_CS_N	PIN_A10	Chip select	3.3V
ADC_SADDR	PIN_B10	Digital data input	3.3V
ADC_SDAT	PIN_A9	Digital data output	3.3V
ADC_SCLK	PIN_B14	Digital clock input	3.3V

## **3.7 Digital Accelerometer**

The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution measurement. This digital accelerometer can be accessed through a SPI 3-wire digital interface or I2C 2-wire digital interface. Main applications include medical instrumentation, industrial instrumentation, personal electronic aid and hard disk drive protection etc. Some of the key features of this device are listed below. For more detailed information, please refer to its datasheet which is available on manufacturer's website or under the /datasheet folder of the system CD.

- Up to 13-bit resolution at +/- 16g
- SPI (3- wire) or I2C (2-wire) digital interface
- Flexible interrupts modes

Figure 3-13 shows the connections between the ADXL345 and the Cyclone IV E device.

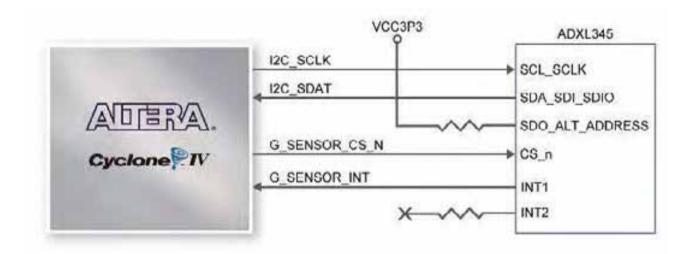


Figure 3-13 Wiring between the ADXL345 and the Cyclone IV E device

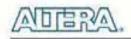
Signal Name	FPGA Pin No.	Description	I/O Standard
I2C_SCLK	PIN_F2	EEPROM clock	3.3V
I2C_SDAT	PIN_F1	EEPROM data	3.3V
G_SENSOR_INT	PIN_M2	G_Sensor Interrupt	3.3V
G_SENSOR_CS_N	PIN_G5	G_Sensor chip select	3.3V

Table 3-10	Pin Assignment	s for Digital /	Accelerometer
1abic 5-10	i in Assignment	S IVI Digital	

## **3.8 Clock Circuitry**

The DE0-Nano board includes a 50 MHz oscillator. The oscillator is connected directly to a dedicated clock input pin of the Cyclone IV E FPGA. The 50MHz clock input can be used as a source clock to drive the phase lock loops (PLL) circuit. The clock distribution on the DE0-Nano board is shown in **Figure 3-14**.

terasiC Terasic DE0-Nano User Manual



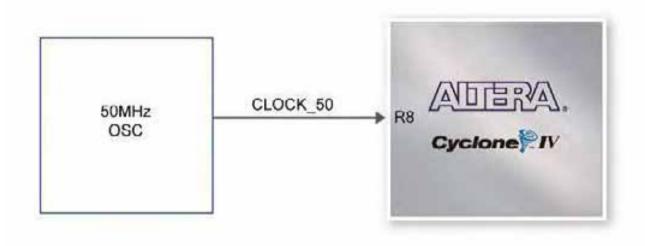


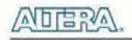
Figure 3-14 Block diagram of the clock distribution

## **3.9 Power Supply**

The DE0-Nano board's power is provided through the USB 5V power, the 5V VCC pins on the two 40-pin headers or the 2-pin power header. The DC voltage is then stepped down to various required voltages. For portable project applications, connect a battery power supply (3.6~5.7V) to the 2-pin external power header shown in **Figure 3-15**.



Figure 3-15 Portable Battery Connection



#### Power Distribution System

Figure 3-16 shows the power distribution system on the DEO-Nano board.

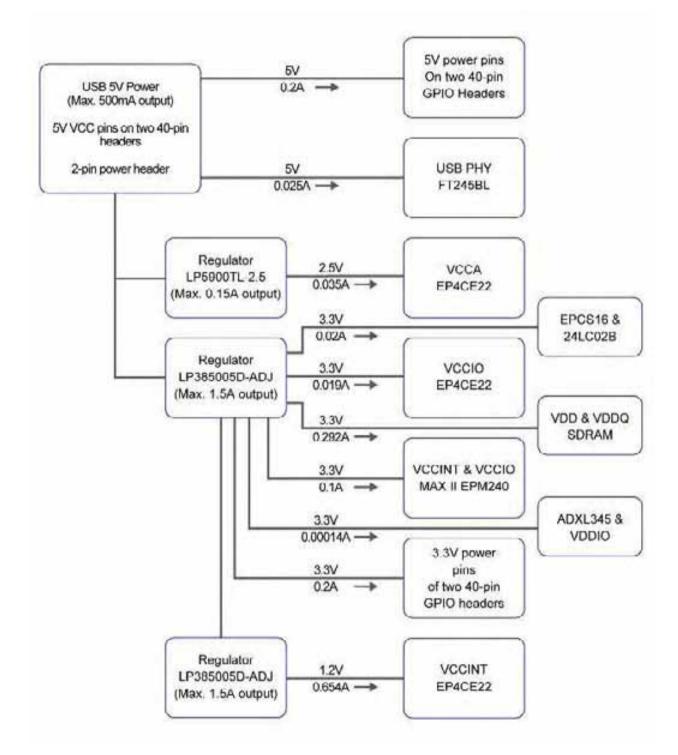
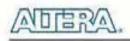


Figure 3-16 DE0-Nano Power Distribution System



## **Chapter 4**

# **DE0-Nano Control Panel**

The DE0-Nano board comes with a Control Panel facility that allows users to access various components on the board from a host computer. The host computer communicates with the board through a USB connection. The facility can be used to verify the functionality of components on the board or be used as a debug tool while developing RTL code.

This chapter first presents some basic functions of the Control Panel, then describes its structure in block diagram form, and finally describes its capabilities.

## **4.1 Control Panel Setup**

The Control Panel Software Utility is located in the directory "*tools/DE0\_NANO\_ControlPanel*" in the **DE0-Nano System CD**. It's free of installation, just copy the whole folder to your host computer and launch the control panel by executing the "DE0\_NANO\_ControlPanel.exe".

When Control Panel starts it will attempt to download a configuration file onto the DEO-Nano board. The configuration file contains a design that communicates with the peripheral devices on the board that are attached to the FPGA device. Perform the following steps to ensure that the control panel starts up successfully:

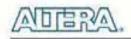
- 1. Make sure Quartus II 10.0 or later version is installed successfully on your PC.
- 2. Connect a USB A to Mini-B cable to a USB (Type A) host port and to the board.

3. Start the executable DE0\_NANO\_ControlPanel.exe on the host computer. The Control Panel user interface shown in **Figure 4-1** will appear.

5. The DE0\_NANO\_ControlPanel.sof bit stream is loaded automatically as soon as the DE0\_NANO\_ControlPanel.exe is launched.

6. In case the connection is disconnected, click on CONNECT where the .sof will be re-loaded onto the board.

Note: the Control Panel will occupy the USB port until you choose to close the program or disconnect it from the board by clicking the Disconnect button. While the Control Panel is connected to the board, you will be unable to use Quartus II to download a configuration file into the FPGA.

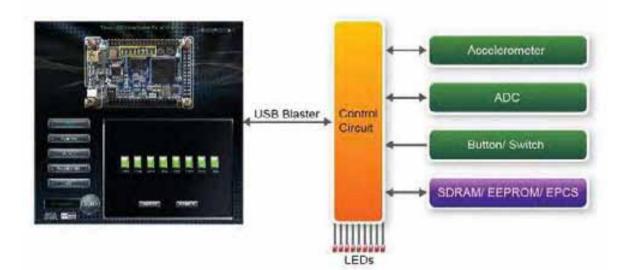


8. The Control Panel is now ready for use; experience it by setting the ON/OFF status for some LEDs and observing the result on the DE0-Nano board.



Figure 4-1 The DE0-Nano Control Panel

The concept of the DEO-Nano Control Panel is illustrated in **Figure 4-2**. The "Control Circuit" that performs the control functions is implemented in the FPGA board. It communicates with the Control Panel window, which is active on the host computer, via the USB Blaster link. The graphical interface is used to issue commands to the control circuit. It handles all requests and performs data transfers between the computer and the DEO-Nano board.





The DE0-Nano Control Panel can be used to light up LEDs, change the buttons/switches status, read/write to SDRAM Memory, read ADC channels, and display the Accelerometer information.

## **4.2 Controlling the LEDs**

A simple function of the Control Panel is to allow setting the values displayed on LEDs. Choosing the **LED** tab displays the window in **Figure 4-3**. Here, you can directly turn the LEDs on or off individually or by clicking "Light All" or "Unlight All".



Figure 4-3 Controlling LEDs

## **4.3 Switches and Pushbuttons**

Choosing the **Switches** tab displays the window in **Figure 4-4**. The function is designed to monitor the status of slide switches and pushbuttons in real time and show the status in a graphical user interface. It can be used to verify the functionality of the slide switches and pushbuttons.





Figure 4-4 Monitoring switches and buttons

The ability to check the status of pushbutton and slider switches is not needed in typical design activities. However, it provides a simple mechanism for verifying if the buttons and switches are functioning correctly. Thus, it can be used for troubleshooting purposes.

## **4.4 Memory Controller**

The Control Panel can be used to write/read data to/from the SDRAM/EEPROM/EPCS on the DE0-Nano board. As an example, we will describe how the SDRAM may be accessed; the same approach is used to access the EEPROM and EPCS. Click on the Memory tab and select "SDRAM" to reach the window in **Figure 4-5**.

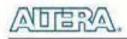




Figure 4-5 Accessing the SDRAM

A 16-bit word can be written into the SDRAM by entering the address of the desired location, specifying the data to be written, and pressing the Write button. Contents of the location can be read by pressing the Read button. **Figure 4-5** depicts the result of writing the hexadecimal value 06CA into offset address 200, followed by reading the same location.

The Sequential Write function of the Control Panel is used to write the contents of a file into the SDRAM as follows:

1. Specify the starting address in the Address box.

2. Specify the number of bytes to be written in the Length box. If the entire file is to be loaded, then a checkmark may be placed in the File Length box instead of giving the number of bytes.

3. To initiate the writing process, click on the Write a File to Memory button.

4. When the Control Panel responds with the standard Windows dialog box asking for the source file, specify the desired file in the usual manner.

The Control Panel also supports loading files with a .hex extension. Files with a .hex extension are ASCII text files that specify memory values using ASCII characters to represent hexadecimal values. For example, a file containing the line

#### 0123456789ABCDEF

defines eight 8-bit values: 01, 23, 45, 67, 89, AB, CD, EF. These values will be loaded consecutively into the memory.

The Sequential Read function is used to read the contents of the SDRAM and fill them into a file as follows:

1. Specify the starting address in the Address box.

2. Specify the number of bytes to be copied into the file in the Length box. If the entire contents of the SDRAM are to be copied (which involves all 32 Mbytes), then place a checkmark in the Entire Memory box.

3. Press Load Memory Content to a File button.

4. When the Control Panel responds with the standard Windows dialog box asking for the destination file, specify the desired file in the usual manner.

Users can use the similar way to access the EEPROM and EPCS. Please note that users need to erase the EPCS before writing data to it.

## **4.5 Digital Accelerometer**

The Control Panel can be used to display the status of the Digital Accelerometer where it measures the output of its 3-axis (X, Y, Z). The measurement range and resolution is set to default value  $\pm 2g$  (acceleration of gravity) and 10bit twos complement respectively. Figure 4-6 shows the current digital accelerometer status of the DE0-Nano when Accelerometer tab is clicked. The units that are displayed are the raw register values converted to decimal. The value in parentheses is the gravitational acceleration values (mg) calculated from the register values according the formula. Table 4-1 shows the rule.

Register Value	*Formula	Result (mg)
0	0/511*2	0
1	1/511*2	3.9
2	2/511*2	6.8
17	17/511*2	66.4
511	511/511*2	2000

 Table 4-1 acceleration values convert rule

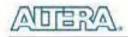




Figure 4-6 Digital Accelerometer status

## 4.6 ADC

From the Control Panel, users are able to view the eight-channel 12-bit analog-to-digital converter reading. The values shown are the ADC register outputs from all of the eight separate channels. The voltage shown is the voltage reading from the separate pins on the extension header. **Figure 4-7** shows the ADC readings when the ADC tab is chosen.



Figure 4-7 ADC Readings

## **4.7 Overall Structure of the DE0-Nano Control Panel**

The DE0-Nano Control Panel is based on a Nios II SOPC system instantiated in the Cyclone IV E FPGA with software running on the on-chip memory. The software part is implemented in C code; the hardware part is implemented in Verilog HDL code with SOPC builder. The source code is not available on the DE0-Nano System CD.

To run the Control Panel, users should make the configuration according to Section 4.1. **Figure 4-8** depicts the structure of the Control Panel. Each input/output device is controlled by the Nios II Processor instantiated in the FPGA chip. The communication with the PC is done via the USB Blaster link. The Nios II interprets the commands sent from the PC and performs the corresponding actions.

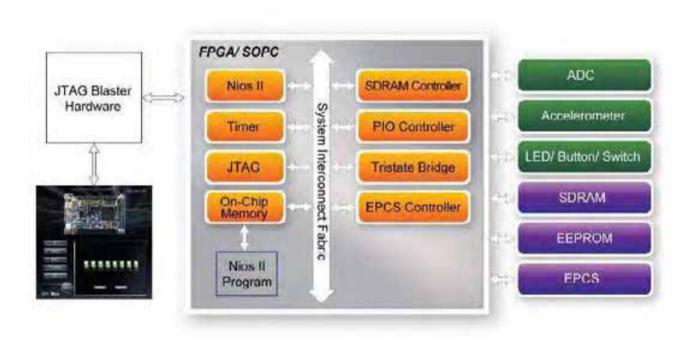
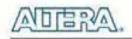


Figure 4-8 The block diagram of the DE0-Nano Control Panel



## Chapter 5

DE0-Nano System Builder

This chapter describes how users can create a custom design project on the DE0-Nano board by using DE0-Nano Tool – DE0-Nano System Builder.

## **5.1 Introduction**

The DE0-Nano System Builder is a Windows based software utility, designed to assist users in creating a Quartus II project for the DE0-Nano board within minutes. The generated Quartus II project files include:

- Quartus II Project File (.qpf)
- Quartus II Setting File (.qsf)
- Top-Level Design File (.v)
- Synopsys Design Constraints file (.sdc)
- Pin Assignment Document (.htm)

By providing the above files, DE0-Nano System Builder helps to prevents occurrence of situations that are prone to errors when users manually edit the top-level design file or place pin assignments. The common mistakes that users encounter are the following:

1. Board damaged for wrong pin/bank voltage assignments.

2. Board malfunction caused by wrong device connections or missing pin counts for connected ends.

3. Performance degeneration because of improper pin assignments.

### **5.2 General Design Flow**

This section will introduce the general design flow to build a project for the DE0-Nano board via the DE0-Nano System Builder. The general design flow is illustrated in **Figure 5-1**.

To create a new system using the DEO-Nano System Builder, begin by launching the DEO-Nano System Builder software. The software will then prompt you to specify the name of the project you wish to create, as well as the components on the DEO-Nano board you wish to you. Once your specification is complete, you can generate the system.



The generated system is described using several files. In particular, there is the project file (.qpf), the top-level Verilog wrapper file (.v) that describes the I/O pins you will use in your design, and the Quartus II settings file (.qsf) that specifies which pin on the FPGA each I/O in your design should connect to. A Synopsys Design Constraints (.sdc) file with timing constraints and an HTML file with pin descriptions will be generated as well.

To proceed with your design, open the Quartus II CAD software and open your newly-created project. You will now be able to implement the logic of your design by describing your design in a hardware description language, and connecting it to I/Os in the top-level wrapper file. Once your design is complete, compile the design using Quartus II, and then use the Quartus II Programmer tool to configure the FPGA on the DEO-Nano board, using the JTAG programming mode.

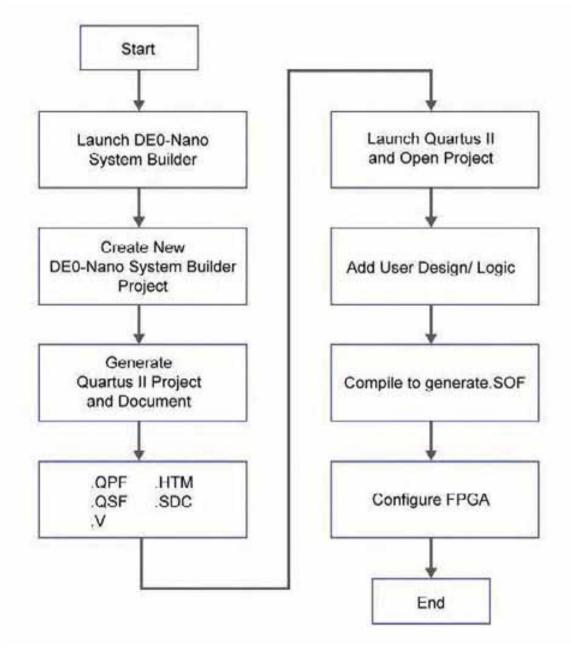


Figure 5-1 The general design flow of building a design

## 5.3 Using DE0-Nano System Builder

This section provides the detailed procedures on how the to use the DEO-Nano System Builder.

#### ■ Install and launch the DE0-Nano System Builder

The DEO-Nano System Builder is located in the directory: "*Tools\DEO\_NANO\_SystemBuilder*" on the DEO-Nano System CD. Users can copy the whole folder to a host computer without installing the utility. Launch the DEO-Nano System Builder by executing the DEO\_NANO\_SystemBuilder.exe on the host computer and the GUI window will appear as shown in Figure 5-2.

rein 1950-Nane System Builder VI	0.0		107
AND TERMS		-System Configuration Project Nome DE9_NANO	
DEO-Nano FPGA	Board	Duttor K 2 Duttor K 2 M SDRAM 22MB DEEPRON, 2Kb DAccelerumeter OPIO, 0 Header	ELEU × 8 E Dip Ovitch x 4 Pladic E EPOS E 2x13 Pin Header
		Nore PreterName OPIO I Header	
Load Setting	Generate	Nare	
Save Setting	Ext	Profix Namo	

Figure 5-2 The DE0-Nano System Builder window

#### Input Project Name

Input project name as show in Figure 5-3.

Project Name: Type in an appropriate name here, it will automatically be assigned as the name of your top-level design entity.



		System Cottligutation Project Name			
ROGEAM		(DEC_NANO			
DE0-Nano FPGA	Board	CLOCK Button x 2 SURAM, 32MB ELEPTION, 2Ks Accelerometer GPIO-0 Header	ULED ×8 Dop Switch x 4 MADC DCPCO PI2x15 Pin Header		
		None Profix Name:	~		
		GPIO-L Header			
Load Setting	Generate	None	2		
Save Setting	Dat	Prefs Name:			

Figure 5-3 The DE0-Nano Board Type and Project Name

#### System Configuration

Under System Configuration users are given the flexibility of enabling their choice of included components on the DEO-Nano as shown in **Figure 5-4**. Each component of the DEO-Nano is listed where users can enable or disable a component according to their design by simply marking a check or removing the check in the field provided. If the component is enabled, the DEO-Nano System Builder will automatically generate the associated pin assignments including the pin name, pin location, pin direction, and I/O standard.

renar DEU-Frankyrnen Deilder 71.)	lio -		
		- System Configuration Project Name DEC NAND	
DE0-Nano FPGA	Board	EFCLOCK Button x 2 PISCRAM, 22MD EEPPIOM, 25b EAccelerometer	ELED x 8 Dip Switch x 4 MADC ERCS E2x13 Pri Header
		GPIO 0 Header None Hers Name	
Loed Setting	Generale	GPIO-I Header	-
Save Setting	Dit	Profis Namo:	

Figure 5-4 System Configuration Group

**GPIO Expansion** 



Users can connect GPIO expansion card onto GPIO header located on the DEO-Nano board as shown in **Figure 5-5**. Select the appropriate daughter card you wish to include in your design from the drop-down menu. The system builder will automatically generate the associated pin assignments including the pin name, pin location, pin direction, and IO standard.

If a customized daughter board is used, users can select "GPIO Default" followed by changing the pin name and pin direction according to the specification of the customized daughter board.

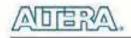
Terreric DED. Roses System Livelity - VI 1	i ii		
MUIERA. UNIVERSITY PROGRAM		System Configuration Project Name DE0_NAKQ	
DEO-Nano FPGA	Board	El CLOCK El Dutton x 2 Pl SOHAM, 32M B EEPROH, 2Ko El Accelerometer OPIO-0 Hender El Table Com Prefix Name	ELEDIx 8 EDip Owthat x 4 PT ALT NT ELEPOS El 2x13 Plat Header
Load Setting	Gunerate	GPIO-1 Header	
Save Setting	Ext	Pretix Name:	1

Figure 5-5 GPIO Expansion Group

The "Prefix Name" is an optional feature which denotes the prefix pin name of the daughter card assigned in your design. Users may leave this field empty.

#### Project Setting Management

The DEO-Nano System Builder also provides functions to restore default setting, loading a setting, and saving users' board configuration file shown in **Figure 5-6**. Users can save the current board configuration information into a .cfg file and load it to the DEO-Nano System Builder.



fernie DEB-Krass System Didder 71.	1.0		(AIP)
ADENA (craste		Rosen Configuration Project Name DEU_NANO	
DE0-Nano FPGA	Contraction of the local division of the loc	CLOCK Duttor x 2 SORAW 32MB EEFROM, 3Kb Accelerometer CPIO-0 Header Official SM Pool Co Prefx Name:	EPCS 2418 Pri Header
Load Setting	Generate	GPIO-I Header	8
Save Setting	Ent	Profis Name:	

Figure 5-6 Project Settings

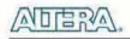
#### Project Generation

When users press the Generate button, the DEO-Nano System Builder will generate the corresponding Quartus II files and documents as listed in the Table 5-1:

Table 5-1The files generated by 1	DE0-Nano System Builder
-----------------------------------	-------------------------

No.	Filename	Description
1	<project name="">.v</project>	Top level Verilog HDL file for Quartus II
2	<project name="">.qpf</project>	Quartus II Project File
3	<project name="">.qsf</project>	Quartus II Setting File
4	<project name="">.sdc</project>	Synopsys Design Constraints file for Quartus II
5	<project name="">.htm</project>	Pin Assignment Document

Users can use Quartus II software to add custom logic into the project and compile the project to generate the SRAM Object File (.sof).



## **Chapter 6**

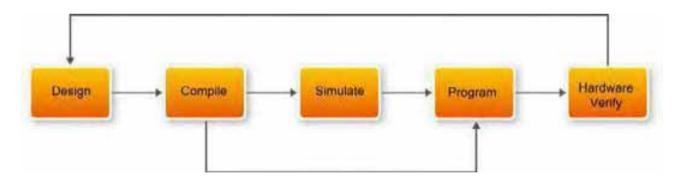
# Tutorial: Creating an FPGA Project

This tutorial provides comprehensive information for understanding how to create a FPGA design and run it on the DEO-Nano development and education board. The following sections provide a quick overview of the design flow, explaining what is needed to get started, and describe what is taught in this tutorial.

### **6.1 Design Flow**

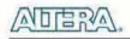
Figure 6-1shows a block diagram of the FPGA design flow.

The first step in the FPGA design flow starts is design entry. The standard design entry methods are using schematics or a hardware description language (HDL), such as Verilog HDL or VHDL. The design entry step is where the designer creates the digital circuit to be implemented inside the FPGA. The flow then proceeds through compilation, simulation, programming, and verification in the FPGA hardware.





This tutorial describes all of the steps except for simulation. Although it is not covered in this document, simulation is very important to learn. There are two types of simulation, Functional and Timing Functional simulation allows you to verify that your hardware is performing the desired functionality. Timing (or post place-and-route) simulation verifies that the design meets timing and functions appropriately in the device. Simulation tutorials can be found on the Altera University Program website at http://university.altera.com.



## 6.2 Before You Begin

This tutorial assumes the following prerequisites

■ You have a general understanding of FPGAs. This tutorial does not explain the basic concepts of programmable logic.

■ You are somewhat familiar with digital circuit design and electronic design automation (EDA) tools.

■ You have installed the Altera Quartus II 10.1 software on your computer. If you do not have the Quartus II software, you can download it from the Altera web site at www.altera.com/download.

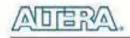
■ You have a DE0-Nano Development Board on which you will test your project. Using a development board helps you to verify whether your design is really working.

■ You have gone through the quick start guide and/or the getting started user guide for your development kit. These documents ensure that you have:

- Installed the required software.
- Determined that the development board functions properly and is connected to your computer.

Next step is to install the USB-Blaster driver, if not already done. To install the driver, connect a USB cable between the DE0-Nano board and a USB port on a computer that is running the Quartus II software.

The computer will recognize the new hardware connected to its USB port, but it will be unable to proceed if it does not have the required driver already installed. If the USB-Blaster driver is not already installed, the New Hardware Wizard in **Figure 6-2** will appear.



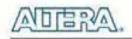


#### Figure 6-2 Found New Hardware Wizard

The desired driver is not available on the Windows Update Web site, therefore select "No, not this time" and click **Next**. This leads to the window in **Figure 6-3**.



Figure 6-3 The driver is found in a specific location



The driver is available within the Quartus II software. Hence, select "Install from a list or specific location" and click **Next** to get to **Figure 6-4**.

Found New Hardware Wizard
Please choose your search and installation options.
<ul> <li>Search for the best driver in these locations.</li> </ul>
Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.
Search removable media (floppy, CD-ROM)
Include this location in the search:
C:\altera\10.1\quartus\drivers\usb-blaster View Browse
O Don't search. I will choose the driver to install.
Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.
< Back Next > Cancel

Figure 6-4 Specify the location of the driver

Now, select "Search for the best driver in these locations" and click Browse to get to the pop-up dialog box in **Figure 6-5** Find the desired driver, which is at location

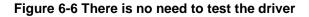
C:\altera\10.1\quartus\drivers\usb-blaster. Click OK and then upon returning to **Figure 6-4** click Next. At this point the installation will commence, but a dialog box in **Figure 6-6** will appear indicating that the driver has not passed the Windows Logo testing. Click Continue Anyway.

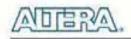


Browse For Folder	? 🔀
Select the folder that contains drivers for your hardwar	e.
⊟ 🛅 10.1 ⊕ 🔂 installer	*
⊞ 🛅 ip ⊞ 🫅 nius2eds	
□ 🔁 guartus ⊞ 🛅 bin ₩ 🔂 bin64	
🕀 🧰 common 🛞 🍋 cusp	
i386 isentinel isertinel isertinel isertinel isertinel isertinel isertinel	
in in dsp_builder In in in dsp_builder	~
To view any subfolders, click a plus sign above.	M
OK	Cancel

Figure 6-5 Browse to find the location







The driver will now be installed as indicated in **Figure 6-7**. Click **Finish** and you can start using the DE0-Nano board.

Found New Hardware Wize	ard
	Completing the Found New Hardware Wizard The wizard has finished installing the software for: Altera USB-Blaster
	< Back Finish Cancel

Figure 6-7 The driver is installed

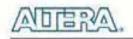
## 6.3 What You Will Learn

In this tutorial you will perform the following tasks:

Create a design that causes LEDs on the development board to blink at two distinct rates. This design is easy to create and gives you visual feedback that the design works. Of course, you can use your DE0-Nano board to run other designs as well. For the LED design, you will write Verilog HDL code for a simple 32-bit counter, add a phase-locked loop (PLL) megafunction as the clock source, and add a 2-input multiplexer megafunction. When the design is running on the board, you can press an input switch to multiplex the counter bits that drive the output LEDs.

## 6.4 Assign The Device

Begin this tutorial by creating a new Quartus II project. A project is a set of files that maintain information about your FPGA design. The Quartus II Settings File (.qsf) and Quartus II Project File (.qpf) files are the primary files in a Quartus II project. To compile a design or make pin assignments, you must first create a project. The steps used to create a project are:



1. In the Quartus II software, select **File > New Project Wizard**. The Introduction page opens, as shown in **Figure 6-8**.

S New Project Winned	8
Introduction	
The New Project Wizard helps you create a new project and preliminary pro Project name and decopery Name of the top-level design entity Wigner files and alreaders Target device family and device Target device family and device	rect settings, including the following:
You can change the settings for an existing project and specify additional p the various pages of the Settings datage box to add functionality to the proj	rotect-wide settings with the Settings command (Assignments menu). You can use local
Don't show me this introduction again	
	a gina (gena canon) (gena

#### Figure 6-8 New Project Wizard introduction

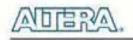
2. Click Next.

3. Enter the following information about your project: (Note: File names, project names, and directories in the Quartus II software cannot contain spaces.)

a. What is the working directory for this project? Enter a directory in which you will store your Quartus II project files for this design. For example, **E:\My\_design\my\_first\_fpga**.

b. What is the name of this project? Type **my\_first\_fpga**.

c. What is the name of the top-level design entity for this project? Type **my\_first\_fpga**. See **Figure 6-9**.

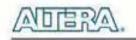


iow Project Witzard	
irectory, Name, Top-Level Entity [page 1 of 5]	
hat is the working decitory for this project?	
(My_designiny_tex_tops	
hat is the name of this project?	
w_finst_faqa	lin
hat is the name of the top-level design entity for this protect? This name is case sensitive and must exactly match the entity name in the design file.	
w_fwrt_fega	-
se Existing Project Settings	

Figure 6-9 Project information

d. Click Next.

e. In the next dialog box, you will assign a specific FPGA device to the design. Select the **EP4CE22F17C6** device, as it is the FPGA on the DE0-Nano, as shown in **Figure 6-10**.



Device family				2how in Available deviced list					
				-					
and a second second	e IV.E			1	Padjage: Pri gounti	Any Any Any		٣	
Devices- Al		_		1					
Target device					Speed grades			-	
	S				Deve adversed devices				
100 Table 100	selected by the Pitter				L PARTE STRATES				
Sedfic devi	ca selected in "Availab	He denoues'	ht		Indexada	contragore (e.g.)			
QONEL HEA									
atable devices:									
Name	Core Voltage	115	User I/Os	Hem	ory Bits	Embedded multiplier 9-bit elements	PLL	100	
P41222277	1.29	22.370	80	640256	13		4	20	
P4CE22E228	1.0V	22220	80	608256	133		4	20	
PHO:22 17A7	\$429	22320	154	608256	133		4	20	
1401221006	11.24	133330	154	608356	111		4	x	
PRE27 1707	1.29	72320	154	600256	13		4	-20	
P4CE22F17C8	1.2V	22320	154	608256	13		4	20	
PREZE IAM.	1.09	32320	354	600756	13		4	20 y	
0			1984					2	
Concession de la	ý.							1.00	
								-	
He d.opy									

Figure 6-10 Specify the Device Example

#### f. Click Finish.

4. When prompted, select **Yes** to create the my\_first\_fpga project directory. You just created your Quartus II FPGA project. Your project is now open in Quartus II, as shown in **Figure 6-11**.



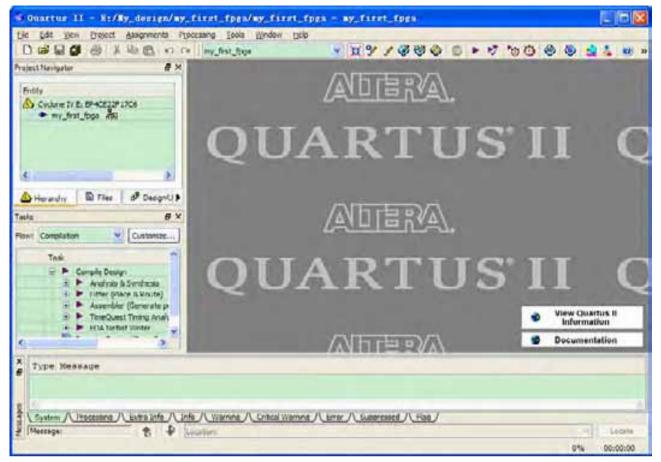


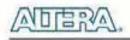
Figure 6-11 my\_first\_fpga project

## **6.5 Creating an FPGA design**

This section describes how to create an FPGA design. This includes creating the top-level design, adding components (in Verilog HDL and using the megafunctions), adding pins and interconnecting all the components and pins.

First, create a top-level module. In this tutorial, you will use schematic entry, via a Block Design File (.bdf). Alternatively, you could use Verilog HDL or VHDL for the top-level module. The following steps describe how to create the top-level schematic.

1. Select File > New > Block Diagram/Schematic File (see Figure 6-12 to create a new file, Block1.bdf, which you will save as the top-level design.



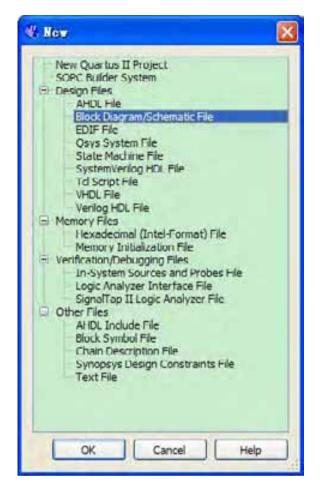


Figure 6-12 New BDF

- 2. Click OK.
- 3. Select **File > Save As** and enter the following information.
  - File name: my\_first\_fpga
  - Save as type: Block Diagram/Schematic File (\*.bdf)
- 4. Click Save. The new design file appears in the Block Editor (see Figure 6-13).



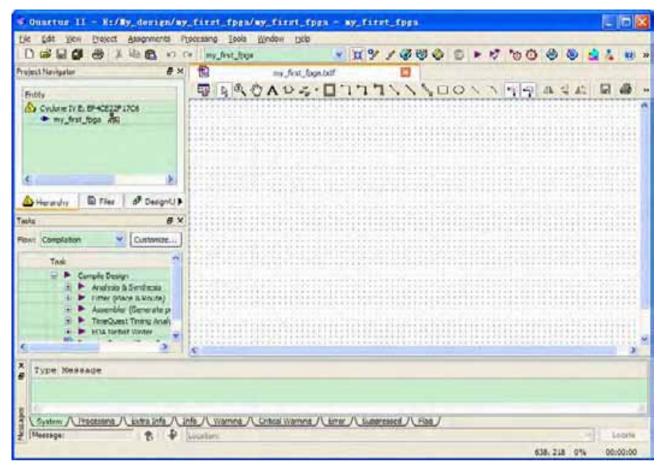


Figure 6-13 Bank BDF

#### Adding a Verilog HDL to the Schematic

- 1. Add HDL code to the blank block diagram by choosing File > New > Verilog HDL File.
- 2. Select Verilog HDL File in the tree and Click OK.

3. Save the newly created file, by selecting **File > Save As** and entering the following information (see **Figure 6-14**).

- File name: simple\_counter.v
- Save as type: Verilog HDL File (\*.v, \*.vlg, \*.verilog)



Save As					? 🔀
Save in: My Recent Documents Desktop My Documents My Computer	my first fpg db greybox_tmp incremental_e counter_bus counter_bus pll.v pll.v simple_counter	dh Jaux.v mux bb.v	• + E		
My Network Places	File name: Save as type:	simple_counter.v       Venlog HDL Files (*.v *.vlg *.ve       I Add file to current project	anlog)	•	Save Cancel

Figure 6-14 Saving the Verilog HDL file

The resulting empty file is ready for you to enter the Verilog HDL code.

4. Type the following Verilog HDL code into the blank simple\_counter.v file, as shown in **Figure 6-15**.

//It has a single clock input and a 32-bit output port

module simple\_counter (

CLOCK\_5,

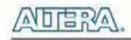
counter\_out

);

input CLOCK\_5;

output [31:0] counter\_out;

reg [31:0] counter\_out;



always @ (posedge CLOCK\_5)

// on positive clock edge

begin

counter\_out <= counter\_out + 1;// increment counter</pre>

end

endmodule

// end of module counter

```
1
      //It has a single clock input and a 32-bit output port
 2
    module simple counter (
 3
                               CLOCK 5,
 4
                               counter out
 5
                              );
 6
      input
                        CLOCK 5 ;
7
      output
                [31:0] counter out;
8
                [31:0] counter out;
      rea
9
10
      always @ (posedge CLOCK 5)
                                                // on positive clock edge
11
    F
         begin
            counter out <= counter out + 1;
12
                                                // increment counter
13
         end
      endmodule
14
                                                // end of module counter
15
```

#### Figure 6-15 The Verilog File of simple\_counter.v

5. Save the file by choosing **File > Save**, pressing **Ctrl + S**, or by clicking the floppy disk icon.

6. Select **File > Create/Update > Create Symbol Files for Current File** to convert the **simple\_counter.v** file to a Symbol File (.sym). You will use this Symbol File to add the HDL code to your schematic.

The Quartus II software creates a Symbol File and displays a message (see Figure 6-16).

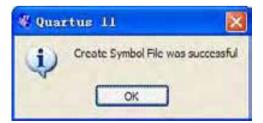


Figure 6-16 Create Symbol File was Successful

- 7. Click OK.
- 8. To add the **simple\_counter.v** symbol to the top-level design, click the **my\_first\_fpga.bdf** tab.

## ADERA.

- 9. Right click in the blank area of the BDF file, and select **Insert > Symbol**.
- 10. Double-click the Project directory to expand it.
- 11. Select the newly created simple\_counter symbol by clicking its icon.

Ubrorica:		*******
🖻 🛱 Project 🔹 🛱 di/altera/10.1/guartus/libraries/	ELICER 5 COLOR ST 10	
si ().		
Nome:		1
smple_counter		
Reveatrise Linude		*******************
Insert symbol as block Launch MesurWaard Plaz In		
MegaWizard Plug-In Manager		*****************

Figure 6-17 Adding the Symbol to the BDF

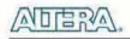
#### 12. Click OK.

13. Move the cursor to the BDF grid; the symbol image moves with the cursor. Click to place the simple\_counter symbol onto the BDF. You can move the block after placing it by simply clicking and dragging it to where you want it and releasing the mouse button to place it. See **Figure 6-18**.

	si	mple_counter		-
		CLOCK_5 counter_out[310]	]	
	ir	Ist		
•		· · · · · · · · · · · · · · · · · · ·		

#### Figure 6-18 Placing the simple\_counter symbol

### terasic DE0-Nano User Manual



14. Press the **Esc key** or click an empty place on the schematic grid to cancel placing further instances of this symbol.

15. Save your project regularly.

#### Adding a Megafunction to the Schematic

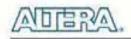
Megafunctions, such as the ones available in the LPM, are pre-designed modules that you can use in FPGA designs. These Altera-provided megafunctions are optimized for speed, area, and device family. You can increase efficiency by using a megafunction instead of writing the function yourself. Altera also provides more complex functions, called MegaCore functions, which you can evaluate for free but require a license file for use in production designs. This tutorial design uses a PLL clock source to drive a simple counter. A PLL uses the on-board oscillator (DE0-Nano Board is 50 MHz) to create a constant clock frequency as the input to the counter. To create the clock source, you will add a pre-built LPM megafunction named ALTPLL.

1. Right click in the blank space in the BDF and select **Insert > Symbol** or click the Add Symbol icon on the toolbar.

2. Click the Megawizard Plug-in Manager button. The MegaWizard<sup>®</sup> Plug-In Manager appears, as shown in **Figure 6-19**.

🧏 Legaliz	ard Plug-In Manager [page 1]	
	The MegaWized Plag-In Manager helps you create or modify design files that contain custom variations of megafunction do you want to perform?	
	Cencel < Back [Next > ] [Init	h

#### Figure 6-19 Mega Wizard Plug-In Manager



- 3. Click Next.
- 4. In MegaWizard Plug-In Manager [page 2a], specify the following selections (see Figure 6-20):

```
a. Select I/O > ALTPLL.
```

b. Under "Which device family will you be using?" select the **Cyclone IV E** for DE0-Nano development board.

c. Under "Which type of output file do you want to create?" select Verilog HDL.

d. Under "What name do you want for the output file?" type pll at the end of the already created directory name.

e. Click Next.

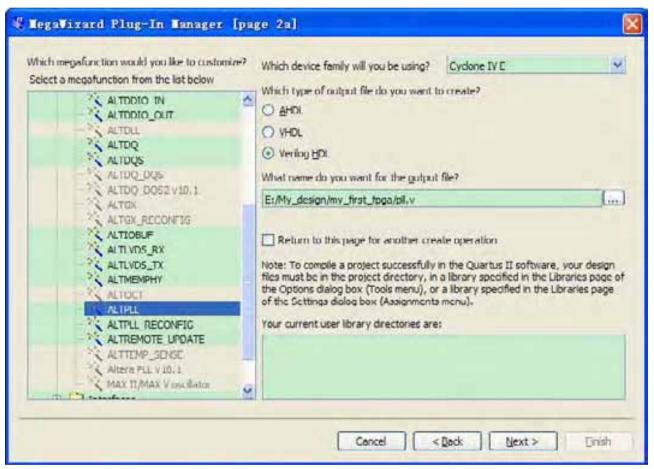
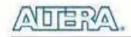


Figure 6-20 MegaWizard Plug-In Manager [page 2a] Selections

5. In the MegaWizard Plug-In Manager [page 3 of 14] window, make the following selections (see Figure 6-21).

- a. Confirm that the currently selected device family option is set to Cyclone IV E.
- b. For device speed grade choose 6 for DE0-Nano.
- c. Set the frequency of the inclock0 input 50 MHz.



#### d. Click Next.

Regalizzard Plug-In Ranages (page 3)	of 14]	2 🛚
ALTPLL		gbout Occumentation
Parameter     Settings     General/Nodes     Saturday: add	(a Damar	
General/Nodes Provide Conductions pli mean analog frequency for according analog for a	Able to implement the requested PLL  Able to implement the requested PLL  Dense al  Which device speed goals will you be using?  Use rathery temperature range devices only  What is the frequency of the includ equil?	Ace family: Circlane: 17 6 Match project/default 50.000 MH/F
	Description	Generation     Generation
	Connation Mode How will the PLL outputs be generated? (a) Use the feedback path inside the PLL (b) <u>On normal mode</u> (c) <u>On normal mod</u> (c) <u>On normal mod</u>	ne)
	Grod	i < Şad (gent > (Bent)

Figure 6-21 MegaWizard Plug-In Manager [page 3 of 14] Selections

6. Unselect all options on MegaWizard page 4. As you turn them off, pins disappear from the PLL block's graphical preview. See **Figure 6-22** for an example.



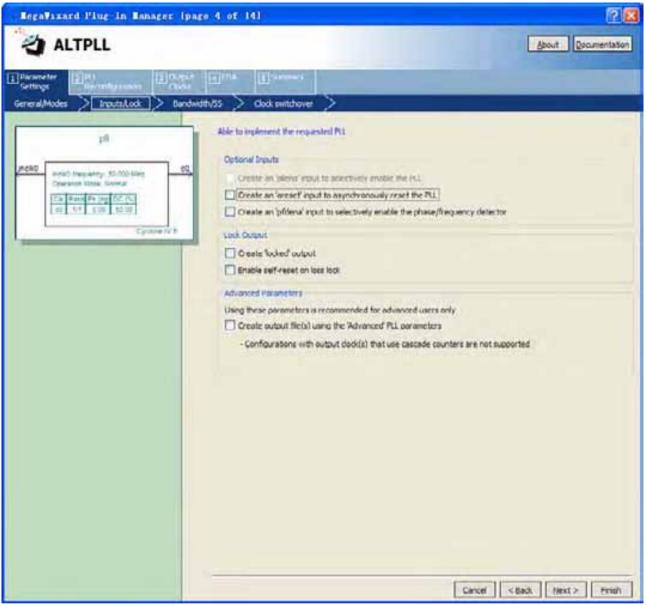


Figure 6-22 MegaWizard Plug-In Manager [page 4 of 14] Selections

- 7. Click **Next** four times to get to page 8.
- 8. Set the Clock division factor to 10, as shown in Figure 6-23.



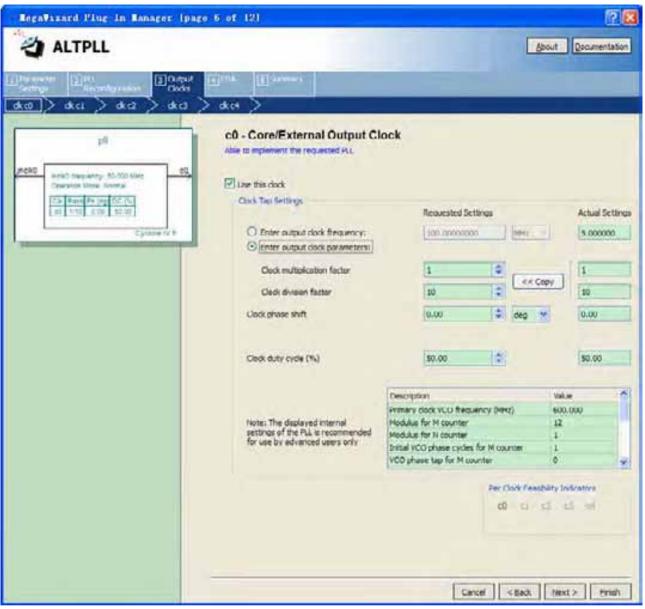
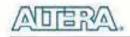


Figure 6-23 MegaWizard Plug-In Manager [page 8 of 14] Selections

9. Click **Next** and then click **Finish**.

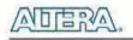
10. The wizard displays a summary of the files it creates (see **Figure 6-24**). Select the pll.bsf option and click Finish again.



Reparatant Plug in Ranager	(page 12 of 12)	Bout Documentation
Elizabete Elizabeteeteeteeteeteeteeteeteeteeteeteeteete		
Sectore 14/80 Departy \$5.000 Mars	discimark indicates maintained in subser	wish to generate. A gray checkmark indicates a file that is automatically generated, and a green an optional file. Click Finish to generate the selected files. The state of each checkbox is guent Megalilitard Pkg-In Manager sessions. g-in Hanager creates the selected files in the following directory: rst_fogol.
Crarpton these through	File	Description
The Party Paring SC 761	2 pl.v	Venisten Rie
at 1.21 0.00 20.00	r plicet	Hwi same/ ports Hit- Ric
Cythree fy h	[]pl.nc	AHOL Indude file
Sector Se	Cipl.orp	VHDL component declaration file
	[]pl.bul	Quartus II symbol file
	pl_mt.v Vpl_tb.v	Instantiation template file Verling HDL blackdoor file
		Cancel (Stack) (mit > ) [ - Emily

Figure 6-24 Wizard-Created Files

The Symbol window opens, showing the newly created PLL megafunction, as shown in Figure 6-25.



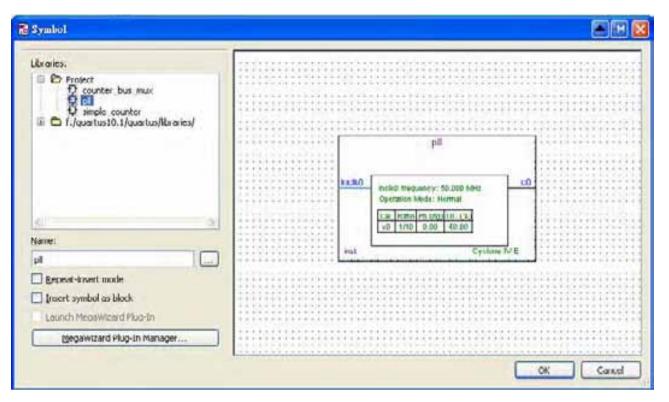


Figure 6-25 PLL Symbol

11. Click **OK** and place the pll symbol onto the BDF to the left of the simple\_counter symbol. You can drag and drop the symbols, if you need to rearrange them. See **Figure 6-26**.

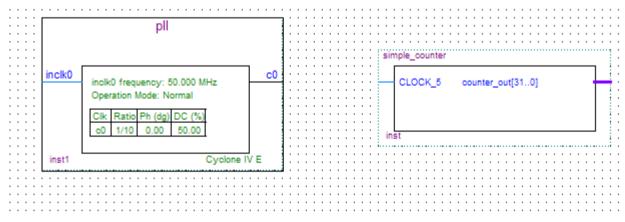
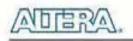


Figure 6-26 Place the PLL Symbol

12. Move the mouse so that the cursor (also called the selection tool) is over the pll symbol's c0 output pin. The orthogonal node tool (cross-hair) icon appears.

13. Click and drag a bus line from the c0 output to the simple\_counter clock input. This action ties the pll output to the simple\_counter input (see **Figure 6-27**).



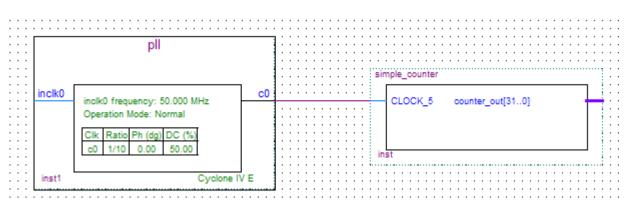


Figure 6-27 Draw a Bus Line connect pll c0 port to simple\_counter CLOCK\_5 port

#### Adding an Input pin to the Schematic

The following steps describe how to add an input pin to the schematic.

- 1. Right click in the blank area of the BDF and select **Insert > Symbol**.
- 2. Under Libraries, select quartus/libraries > primitives > pin >input. See Figure 6-28
- 3. Click OK

If you need more room to place symbols, you can use the vertical and horizontal scroll bars at the edges of the BDF window to view more drawing space.

Symbol.	
Ubrarics: D Project U pl ff simple_counter U di/attera/10.1/quartus/libraries/ di/attera/10.1/quartus/libraries/ di/attera/10.1/quartus/libraries/ di/attera/10.1/quartus/libraries/ D others U primitives + C buffer + C buffer + C buffer + C other D pin ff input ff input ff output + C storage	
Kome: Input Reseathinselt mode Insert nymbol ac block Lawndi MegaWicard Plog In MegaWicard Plog In	

Figure 6-28 Input pin symbol



4. Place the new pin onto the BDF so that it is touching the input to the pll symbol.

5. Use the mouse to click and drag the new input pin to the left; notice that the ports remain connected as shown in **Figure 6-29**.

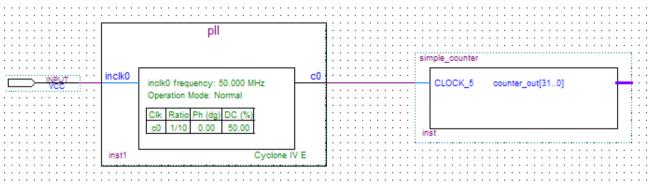


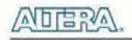
Figure 6-29 Connecting the PLL symbol and Input port

6. Change the pin name by double-clicking pin\_name and typing CLOCK\_50 (see **Figure 6-30**). This name correlates to the oscillator clock that is connected to the FPGA.

#### ■ Adding an Output bus to the Schematic

The following steps describe how to add an output bus to the schematic.

1. Using the Orthogonal Bus tool, draw a bus line connected on one side to the simple\_counter output port, and leave the other end unconnected at about 4 to 8 grid spaces to the right of the simple\_counter.



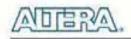
Pin Pr	opert	94	2
General	Forma		
To creati (For exa	e multiple mple: "n	pins, enter a name in AHDL bus notation ne[30]"), or enter a comma-seperated list of name	5,
Bin name	:(s):	100K_50	
Default	alue:	CC	*
-			OK Cancel Help

Figure 6-30 Change the input port name

2. Right-click the new output bus line and select **Properties**.

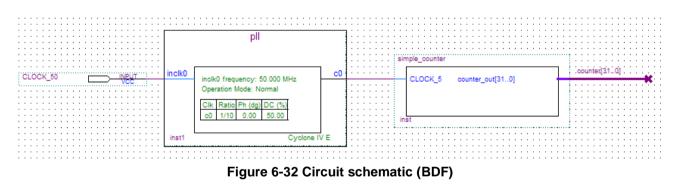
3. Type counter [31..0] as the bus name (see **Figure 6-31**). The notation [X ..Y] is the Quartus II method for specifying the bus width in BDF schematics, where X is the most significant bit (MSB) and Y is the least significant bit (LSB).

4. Click OK. Figure 6-32 shows the BDF.



🖀 Dus Properties	
General Font Format	
Name: counter[010]	
📋 Hide name in block design file.	
	OK Cancel Help

Figure 6-31 Change the output BUS name



#### ■ Adding a Multiplexer to the Schematic

This design uses a multiplexer to route the simple\_counter output to the LED pins on the DEO-Nano development board. You will use the MegaWizard Plug-In Manager to add the multiplexer, lpm\_mux. The design multiplexes two portions of the counter bus to four LEDs on the DEO-Nano board. The following steps describe how to add a multiplexer to the schematic.

## ADERA.

- 1. Right click in the blank area of the BDF and select **Insert > Symbol**.
- 2. Click Megawizard Plug-in Manager.
- 3. Click Next.
- 4. Select Installed Plug-Ins > Gates > LPM\_MUX.

5. Select the **Cyclone IV E** device family, **Verilog HDL** as the output file type, and name the output file **counter\_bus\_mux.v**, as shown in **Figure 6-33**.

6. Click Next.

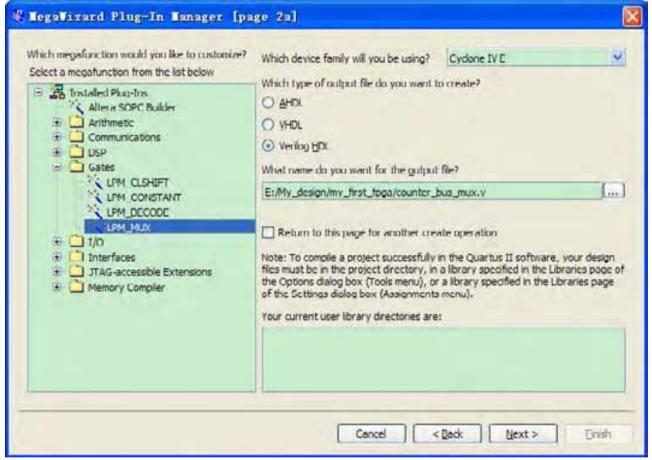
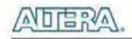


Figure 6-33 Selecting Ipm\_mux

7. Under "How many 'data' inputs do you want?" select 2 inputs (default).

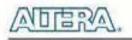
8. Under "How wide should the 'data' input and the 'result' output buses be?" select 4, as shown in **Figure 6-34**.



. MegaWixard Ping-In Manager [pag	e 3 of 5] 🛛 🤇
LPM_MUX	About Documentation
1 Parameter 2 EUA 3 Eluminary Settings	
sei       resutts.ut         data0x[3.0]       resutts.ut	Currently selected device family: Cyclone ti/ E Matul project/default How many 'data' inputs do you want? 2 How wide should the 'data' input and the 'result' output buses be? Do you want to pocine the multiplexer? 0 Yee, I want an output latency of 1 dock cycles Create a Dock Enable input Create a Dock Enable input
Recource Usage 1 lpm_mux	Cancel < gack Next > Enish

Figure 6-34 lpm\_mux settings

- 9. Click Next.
- 10. Click Next.
- 11. Select the **counter\_bus\_mux.bsf** option.
- 12. Click **Finish**. The Symbol window appears (see **Figure 6-35** for an example).



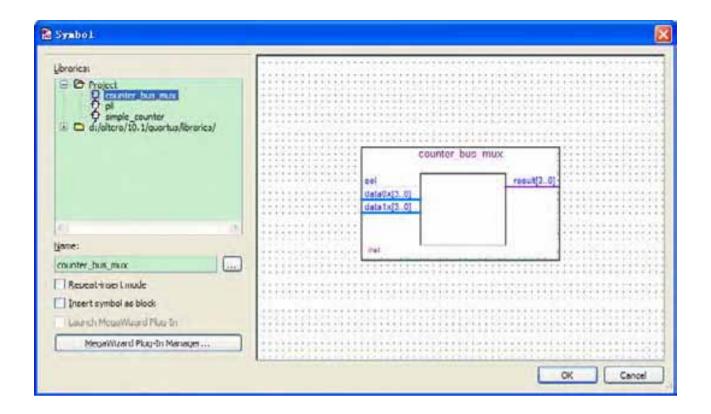
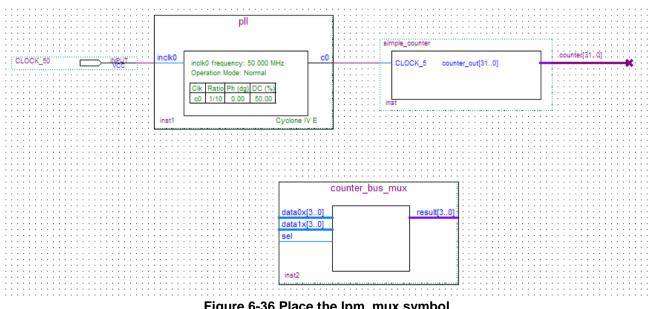


Figure 6-35 lpm\_mux Symbol

#### 13. Click OK

14. Place the counter\_bus\_mux symbol below the existing symbols on the BDF, as shown in Figure 6-36.



## ADERA.

15. Add input buses and output pins to the counter\_bus\_mux symbol as follows:

a. Using the Orthogonal Bus tool, draw bus lines from the data1x[3..0] and data0x[3..0] input ports to about 8 to 12 grid spaces to the left of counter\_bus\_mux.

b. Draw a bus line from the result [3..0] output port to about 6 to 8 grid spaces to the right of counter\_bus\_mux.

c. Right-click the bus line connected to data1x[3..0] and select **Properties**.

d. Name the bus counter[26..23], which selects only those counter output bits to connect to the four bits of the data1x input.

Because the input busses to counter\_bus\_mux have the same names as the output bus from simple\_counter, (counter[x .. y]) the Quartus II software knows to connect these busses.

- e. Click OK.
- f. Right-click the bus line connected to data0x[3..0] and select **Properties**.

g. Name the bus counter [24..21], which selects only those counter output bits to connect to the four bits of the data1x input.

h. Click OK. Figure 6-37 shows the renamed buses.

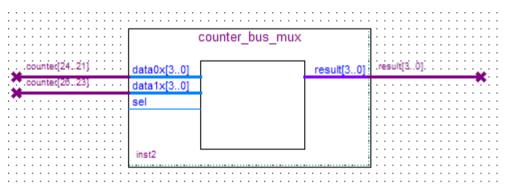


Figure 6-37 Renamed counter\_bus\_mux Bus Lines

If you have not done so already, you may want to save your project file before continuing.

- 16. Right click in the blank area of the BDF and select **Insert > Symbol**.
- 17. Under Libraries, select quartus/libraries > primitives > pin >output, as shown in Figure 6-38.

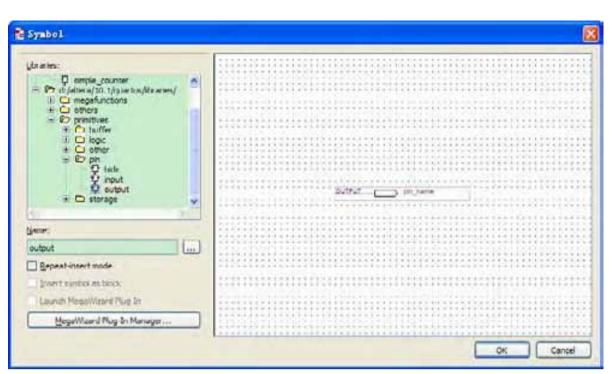


Figure 6-38 Choose output pin

- 18. Click OK.
- 19. Place this output pin so that it connects to the counter\_bus\_mux's result [3..0] bus output line.
- 20. Rename the output pin as LED [3..0]. (see Figure 6-39).

		· · · · · · · · · · · · · · · ·			•••	· ·	· · · ·	· · · ·	  	· · ·	· · ·	· · · ·	· · · ·	· · · ·	· · · · · ·
	C	ounter_bus_mux			• •		• •	• •	 		· · ·	•••			
.counter[2421]	data0x[30]		result[30]				 1		 			• •			01
	data1x[30]								<u>.</u>						
•	sel				• •	• •	• •	· ·	 			• •		• •	
													· · ·		
					• •	• •	• •	· ·	 			• •		• •	
	inst2														
	· · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · ·	· · · · · · · · · · ·	· ·	· · · ·	: :	· ·	· ·	  · · ·		· · ·		· · ·		

#### Figure 6-39 Rename the output pin

- 21. Attach an input pin to the multiplexer select line using an input pin:
- a. Right click in the blank area of the BDF and select **Insert > Symbol**.
- b. Under Libraries, double-click quartus/libraries/ > primitives > pin > input.
- c. Click OK.
- 22. Place this input pin below **counter\_bus\_mux**.
- 23. Connect the input pin to the **counter\_bus\_mux** sel pin.
- 24. Rename the input pin as KEY [0] (see Figure 6-40).

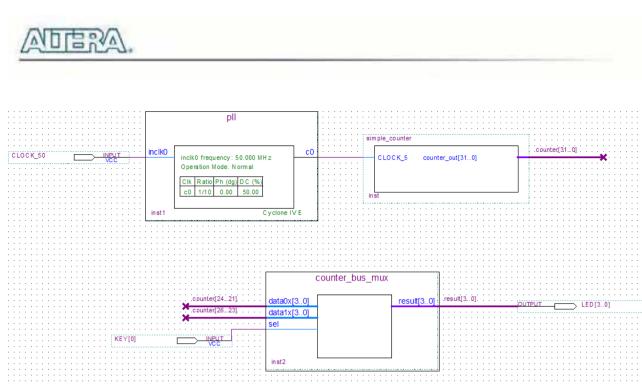


Figure 6-40 Adding the KEY [0] Input Pin

You have finished adding all required components of the circuit to your design. You can add notes or information to the project as text using the Text tool on the toolbar (indicated with the A symbol). For example, you can add the label "OFF = SLOW, ON = FAST" to the KEY [0] input pin and add a project description, such as "DE0-Nano Tutorial Project."

### **6.6 Assign the Pins**

In this section, you will make pin assignments. Before making pin assignments, perform the following steps:

1. Select **Processing > Start > Start Analysis & Elaboration** in preparation for assigning pin locations.

2. Click **OK** in the message window that appears after analysis and elaboration completes.

To make pin assignments to the KEY [0] and CLOCK\_50 input pins and to the LED[3..0] output pins, perform the following steps:

1. Select **Assignments > Pin Planner**, which opens the Pin Planner, a spreadsheet-like table of specific pin assignments. The Pin Planner shows the design's six pins. See **Figure 6-41** 



1000	nedi * M		**					
	Node Name KEY(00) UED[30] Conex groups a	Direction Input Group Output Group	Location		-			
× ®	Named. * ~	os Edu X √	-				Filter Prot al	
6	Node Name	Direction	Location -	L/D filerik	WHEF GROUP	1/O Standard	Reserved	
	mb         KEV[0]           KEV[0]         KEV[0]           KEV[0]         KEV[0]	input Input Output Output Output Output				2.5 v (defaut) 2.5 v (defaul) 2.5 v (defaul) 2.5 v (defaul) 7.5 v (defaul) 7.5 v (defaul) 7.5 v (defaul)		

Figure 6-41 Pin Planner Example

2. In the Location column next to each of the six node names, add the coordinates (pin numbers) as shown in **Table 6-1** for the actual values to use with your DE0-Nano board.

Pin Name	FPGA Pin Location
KEY[0]	J15
LED[3]	A11
LED[2]	B13
LED [1]	A13
LED [0]	A15
CLOCK_50	R8

#### **Table 6-1 Pin Information Setting**

Double-click in the Location column for any of the six pins to open a drop-down list and type the location shown in the table. Alternatively, you can select the pin from a drop-down list. For example, if you type **F1** and press the **Enter** key, the Quartus II software fills in the full PIN\_F1 location name for you. The software also keeps track of corresponding FPGA data such as the I/O bank and VREF Group. Each bank has a distinct color, which corresponds to the top-view wire bond drawing in the upper right window, as shown in **Figure 6-42**.



-143	med: *		# x		-Ann	Contraction of the local division of the loc		
	Node Name GP LED(2) GP LED(1) GP LED(0) Comes anni 0.55	Direction Output Output Output	Locas ***					
×	Named -	ca Edit X V	-				Pitters Prost al	
0	Node Name	Direction	Location	1/CI filenk	WHEF GROUP	UC Standard	Reserved	
P	10 CLOCK 50	Input	PDUKB	1720	BE NO	2.5 V (default)		
ł.	LUP KEYTOT	Input	PD1_318	1	85,10	2.5 V (default)		
	10 LED[2]	Output	PIN_A11	7	\$7_10	2.5 V (default)		
1	🗢 LED[3]	Output	FD1_613	7	87_N0	2.5 V (default)		
	ittp[1]	Dutput	P01_A13	7	87,10	2.5 V (default)		
	<> isbjąj	(3.dps.d	PIN_A15	1	87_145	2.5 V (default)		
	<-mey node>>		1					

Figure 6-42 Completed Pin Planning Example

Now, you are finished creating your Quartus II design!

# 6.7 Create a Default TimeQuest SDC File

Timing settings are critically important for a successful design. For this tutorial you will create a basic Synopsys Design Constraints File (.sdc) that the Quartus II TimeQuest Timing Analyzer uses during design compilation. For more complex designs, you will need to consider the timing requirements more carefully.

To create an SDC, perform the following steps:

- 1. Open the TimeQuest Timing Analyzer by choosing **Tools > TimeQuest Timing Analyzer**.
- 2. Select **File > New SDC file**. The SDC editor opens.
- 3. Type the following code into the editor:

create\_clock -period 20.000 -name CLOCK\_50

derive\_pll\_clocks

derive\_clock\_uncertainty

4. Save this file as my\_first\_fpga.sdc (see Figure 6-43)



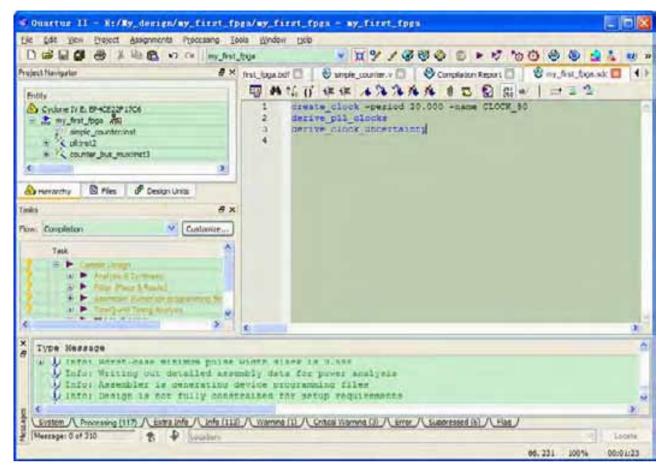


Figure 6-43 Default SDC

Naming the SDC with the same name as the top-level file causes the Quartus II software to use this timing analysis file automatically by default. If you used another name, you would need to add the SDC to the Quartus II assignments file.

## **6.8 Compile Your Design**

After creating your design you must compile it. Compilation converts the design into a bitstream that can be downloaded into the FPGA. The most important output of compilation is an SRAM Object File (.sof), which you use to program the device. Also, the software generates report files that provide information about your circuit as it compiles.

Now that you have created a complete Quartus II project and entered all assignments, you can compile the design.

In the **Processing** menu, select **Start Compilation** or click the **Play** button on the toolbar.

If you are asked to save changes to your BDF, click Yes.

While compiling your design, the Quartus II software provides useful information about the compilation, as shown in **Figure 6-44**.



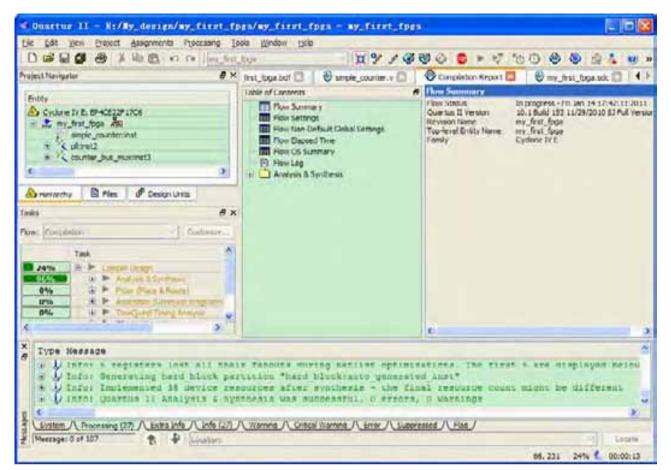


Figure 6-44 Compilation Message for project

When compilation is complete, the Quartus II software displays a message. Click OK to close the message box.

The Quartus II Messages window displays many messages during compilation. It should not display any critical warnings; it may display a few warnings that indicate that the device timing information is preliminary or that some parameters on the I/O pins used for the LEDs were not set. The software provides the compilation results in the Compilation Report tab as shown in **Figure 6-45**.



Flow Summary	
Elow Status         Quartus II Version         Revision Name         Top-level Entity Name         Family         Device         Timing Models         Total logic elements         Dedicated logic registers         Total pins         Total virtual pins         Total memory bits         Embedded Multiplier 9-bit elements         Total PLLs	Successful - Fri Jan 14 17:42:39 2011 10.1 Build 153 11/29/2010 SJ Full Version my_first_fpga Cyclone IV E EP4CE22F17C6 Final 31 / 22,320 ( < 1 % ) 37 / 22,320 ( < 1 % ) 27 6 / 154 ( 4 % ) 0 0 / 608,256 ( 0 % ) 0 / 132 ( 0 % ) 1 / 4 ( 25 % )

Figure 6-45 Compilation Report Example

## **6.9 Program the FPGA Device**

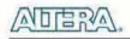
After compiling and verifying your design you are ready to program the FPGA on the development board. You download the SOF you just created into the FPGA using the USB-Blaster circuitry on the board. Set up your hardware for programming using the following steps:

First, connect the USB cable, which was included in your development kit, between the DEO-Nano and the host computer. Refer to the getting started user guide for detailed instructions on how to connect the cables.

Refer to the getting started user guide for detailed instructions on how to connect the cables.

Program the FPGA using the following steps.

1. Select Tools > Programmer. The Programmer window opens, as shown in Figure 6-46.



🕲 Programmer – 1	E=/Wy_design/wy_	first_fpga/ay_	first_fpga	- <b>ny</b> first	figa - Lu	y [	
File Eqit New Nov	cessing Loois Windo	W					
1 lardware Setup	USD-Diaster [USD-0]	Mode: JTAG	D.	Prog	ress:		
Enable real-time ISP	to allow background progr	ramming (for MAX II and	MAX V devices)				
Start	nie	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check
M <sup>11</sup> Stop	my first fpoal.sof	EP40222F17	00138880	FITTFIFF	2		
Auto Detect							
X Delete							
Car and File							
Change Hie	<						
Saue File							6
Add Device							
1ª Up	A						
- Devin							
	B						
	FP4CF7	2F17					
	. 100						×

Figure 6-46 Programmer Window

2. Click Hardware Setup.

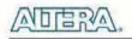
3. If it is not already turned on, turn on the USB-Blaster [USB-0] option under currently selected hardware, as shown in **Figure 6-47**.



ect a programming dware setup appli					s. This programming
mently selected have	ardware:	USB-Blast No Hardy	er (US8-0)		
Hardware			Server	Port	Add Hardware
USB-Blaster			Local	USB-0	Remove Hardware

Figure 6-47 Hardware Setting

- 4. Click Close.
- 5. If the file name in the Programmer does not show **my\_first\_fpga.sof**, click **Add File**.
- 6. Select the my\_first\_fpga.sof file from the project directory (see Figure 6-48).
- 7. Click the **Start** button.



	Dr/Home/User/Deskto Processing Tools Wind	and the second se		Proge	_	(Surcessfu	
A STATE OF A	SP to allow background pro	gramming (for MAX II arv	d MAX Y devices)				
Mu Start	File	Device	Checkoum	Usercode	Program/ Configure	Verify	Hlat
#h 200	my_first_fpas.sof	EP4CE22F17	00107CD0	FFFFFFFF		EU	
Auto Detext							
X Delete							
Add File							
Change File							
Save File	e						2
Add Device							
Pup	S. Contra	CODE T					
E Down		BRØAL					
	1.5 1 1.6 1	10 m					

Figure 6-48 Downloading Complete

Congratulations, you have created, compiled, and programmed your first FPGA design! The compiled SRAM Object File (.sof) is loaded onto the FPGA on the development board and the design should be running.

## **6.10 Verify The Hardware**

When you verify the design in hardware, you observe the runtime behavior of the FPGA hardware design and ensure that it is functioning appropriately.

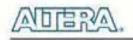
Verify the design by performing the following steps:

1. Observe that the four development board LEDs appear to be advancing slowly in a binary count pattern, which is driven by the simple\_counter bits [26..23].

The LEDs are active low, therefore, when counting begins all LEDs are turned on (the 0000 state).

2. Press and hold KEY [0] on the development board and observe that the LEDs advance more quickly. Pressing this KEY causes the design to multiplex using the faster advancing part of the counter (bits [24..21]).

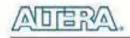
3. If other LEDs emit faintness light, select Assignments > Device. Click Device and Options. See Figure 6-49.



Deuce family				Show in 'Available deviced' list			
Eamly: Cydore	IVE		×	Package:	Arty	1	
Devices: Al				Pin count:	Any	×	
Desires lie					1.1.1.		
Target device				Speed grade:	Any		
O Auto desire	selected by the Fit	÷		Show adve	anced devices		
1007 (1002) 2 (10)	ce celected in 'Ava		let	LingdCopy	compatible only		
Otheri n/a		TATACIANCE					
				Device and Pin	Options		
Available devices:							
Name	Core Voltage	e IFs	User I/Os	Memory Bits	Embedded multiplier 9-bi	t elements 📫	
EP4CE72E22C9L	1.0V	22320	80	608256	132		
EP-ICE22E2217	1.2V	22320	80	608256	132		
EP4CE22E2218L	1.0V	22320	80	608256	132		
EP4CE23F17A7	1.29	22320	154	608256	132	_	
EP4CE22F17C6	1.27	22320	154	608256	1.52		
EP4CE22F17C7 EP4CE22F17C8	1.2V 1.2V	22320	154	608256	102		
EP4CE22F17C8	1.07	22320	154	608256	132		
EP4CE22F17C9L	1.0V	22320	154	608256	132		
CP +CE22F 1717	1.29	22320	154	608256	132		
FP4CF22F1718	1.01	22320	154	608256	137		
	1.74	22320	174	A0474A	117	*	
4				- 100 - 100		\$	
Moraban campa	tbity C	omponion des	lice				
Migration De	nices H	GrdCopy:				1	
-			a state of the local division of the local d	device resources			

Figure 6-49 Device and Options

Select unused pins. Reserve all unused pins: select the As input tri-stated option. See Figure 6-50.

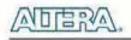


♥ Device and Pin Options - my	_firat_fpga 🔣
Celegory:	
General Configuration	Unused Pins
Programming Files Unused Pins Dual Purpose Pins Capacitive Loading	Specify device-wide options for reserving all unused pins on the device. To reserve individual dual-purpose configuration pins, go to the Dual-Purpose Pins tab. To reserve other pins individually, use the Assignment Frittor.
Board Trace Model I/O Timing Voltage Pin Placement Error Detection CRC CvPCIe Settinos	Reserve al grussed pins: As input bi-stated
	Description:
	Reserves all unused pins on the target device in one of 5 states: as inputs that are tri- stated, as outputs that drive ground, as outputs that drive an unspecified signal, as input tri stated with bus hold, or as input tri stated with weak pull up.
	Reset
	OK Cancel Help

Figure 6-50 Setting unused pins

Click twice OK.

4. In the Processing menu, choose **Start Compilation**. After the compile, select **Tools** > **Programmer**. Select the **my\_first\_fpga.sof** file from the project directory. Click **Start**. At this time you could find the other LEDs are off.



# Chapter 7

# Tutorial: Creating a Nios II Project

This tutorial provides comprehensive information that will help you understand how to create a microprocessor system on your FPGA development board and run software on it. This system will be based on the Altera Nios II processor.

# **7.1 Required Features**

This tutorial requires the Quartus II and Nios II EDS software to be installed. The tutorial was written for version 10.1 of those software packages. If you are using a different version, there may be some difference in the flow. Also, this tutorial requires the DEO-Nano board.

# **7.2 Creation of Hardware Design**

This section describes the flow of how to create a hardware system including a Nios II processor.

1. Launch Quartus II then select **File > New Project Wizard**, start to create a new project. See **Figure 7-1** and **Figure 7-2**.



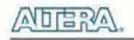
<b>%</b> (	Qu	artus	II			
Fil	.e	<u>E</u> dit	<u>V</u> iew	<u>P</u> roject	<u>A</u> ssignments	P <u>r</u> o
$\square$	Ne	w			Ctrl+N	
<b>2</b>	O₽	en			Ctrl+0	
	<u>C</u> 1	ose			Ctrl+F4	
2	Ne	w Proje	ct <u>W</u> iz:	ard		
ŝ	0p	en P <u>r</u> oj	ect		Ctrl+J	
	Sa	ve Proj	ec <u>t</u>			
	C1	os <u>e</u> Pro	ject			
	Sa	ve			Ctrl+S	
	Sa	ve <u>A</u> s				
ø	Sa	ve All			Ctrl+Shift	+S
	<u>F</u> i	le Prop	erties			
	Cr	eate /	Update			•
		port <u>.</u>				
	Co	nvert P	rogram	ming Files	i	
	Pa	ge Set <u>u</u>	шр			
à	Pr	int Pre	view			
6	<u>P</u> r	int			Ctrl+P	
	Re	cent F <u>i</u>	les			•
	Re	cent Pr	ojects			•
	Ex	it			Alt+F4	

Figure 7-1 Start to Create a New Project

Prectory, Name, Top-Level Entity [page 1 of 5]	
hat is the working directory for this protect?	-
h Y-ovie User (Decktop	1.0
fiel a the same of this grapes?	-
	111
that is the name of the populated design entity for this project? This name is case sensitive and must exactly match the entity name in the design Re-	
	111
In a Existing Project Settings	

Figure 7-2 New Project Wizard

2. Select a working directory for this project, type project name and top-level entity name as shown in **Figure 7-3**. Then click **Next**, you will see a window as shown in **Figure 7-4**.



New Project Vinned	
Directory, Name, Top-Level Entity [page 1 of 5]	
What is the working directory for this project?	
Di, Iny Rest, most	(III)
Wath the name of the groject?	3.4
nifirst_nosi	
that is the name of the pop-level design entity for the project? This name is case sensitive and must exactly match the en	bity name in the design file.
nyfrat_noal	
Lee Existing Project Settings	
clash pert > Print	Carvel Belo

Figure 7-3 Input the working directory, the name of project, top-level design entity

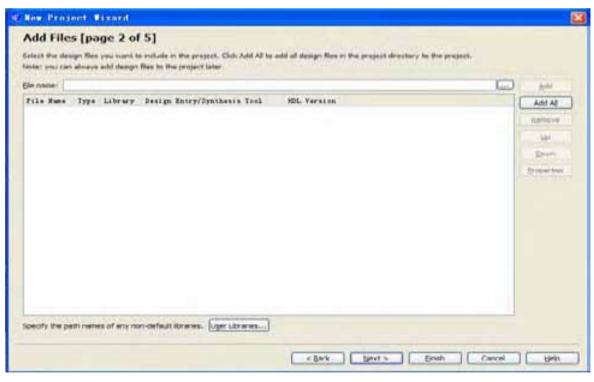
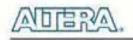


Figure 7-4 New Project Wizard: Add Files [page 2 of 5]

3. Click **Next** to skip in **Add Files** window. In the **Family & Device Settings** window, we will choose device family and device settings appropriate for the DE0-Nano board. You should choose settings the same, as shown in **Figure 7-5**. Then click **Next** to get to the window as shown in **Figure 7-6**.

terasic Terasic DEO-Nano User Manual



	d device you what to tar	yes nor com	hundrer.				
Desize family				Show in 'Avail	ible devices' list		
Lamily: Cydone I	WE.			Packager	LArry	1	
Devites M				Procent	An	~	
a submer lie				Sector Concerns	No. 10		
Target dense				Suged grade:	Any	8	
O Auto device a	elected by the Fitter			W Stress achi	arcet devices		
	e selected in Woolable de			PeriCary coupeling only			
	e selectes in Avaluable un	nices int		( networks	Contraction of AA		
(2thers w/k							
unishie devices.							
	Core Voltage	17.	Buer 1/0x	Benery Bits	Tabaddad aultiplic	or 9-bit elements "	
IP4CECESTIOL	1.07	122320	00	600256	132		
	1.29	72320	154	108256	132		
T-4CE22717A7				608256	132		
1940122711747 1940122711768	1.27	22320	154				
194012271747 194012271765 194012271765	1.27	111111	354	600056	132		
124CE2271747 124CE2271765 124CE2271767 124CE2271768	1.2V 1.EV 1.2V	22320 22320	154 194	600056 N08256	132 132		
174CE2271TAT 174CE2271TAT 174CE2271TCT 174CE2271TCT 174CE2271TCD 174CE2271TCD	1.2V 1.2V 1.2V 1.0V	22320 22320 22320	154 194 154	600056 608256 608258	130 130 132		
EPACE22711AT EPACE22711AT EPACE22711C8 EPACE22711C8 EPACE22711C8L EPACE22711C8L EPACE22711C8L	1.2V 1.2V 1.2V 1.0V 1.0V	22380 22380 22380 22380	154 154 154 154	600056 000256 608258 608256	132 132 132 132		
THCE2271TAT THCE2271TAT THCE2271TCT THCE2271TCD THCE2271TCD THCE2271TCD THCE2271TCD THCE2271TCD	1.2V 1.2V 1.2V 1.0V	22320 22320 22320	154 194 154	600056 608256 608258	130 130 132		
EPACE2271TAT IPACE2271TAT IPACE2271TCT EPACE2271TCD EPACE2271TCDL EPACE2271TCDL EPACE2271TCDL EPACE2271TT	1.2V 1.2V 1.2V 1.0V 1.0V	22380 22380 22380 22380	154 154 154 154	600056 000256 608258 608256	132 132 132 132	,	
ENCE22711A7 INCREMENTATION ENCERCENTION ENCERCENTION ENCERCENTION ENCERCENTION ENCERCENTION	1 2V 1 2V 1 2V 1 2V 1 0V 1 0V 1 2V	22380 22380 22380 22380	154 154 154 154	600056 000256 608258 608256	132 132 132 132	×	
124CE2271747 124CE2271765 124CE2271767 124CE2271768	1 2V 1 2V 1 2V 1 2V 1 0V 1 0V 1 2V	22380 22380 22380 22380	154 154 154 154	600056 000256 608258 608256	132 132 132 132	*	

Figure 7-5 New Project Wizard: Family & Device Settings [page 3 of 5]

4. Click **Next** and will see a window as shown in **Figure 7-7**. **Figure 7-7** is a summary about the new project. Click **Finish** to complete the New Project Wizard. **Figure 7-8** show the new project.



DA tosle:			
fuel Type	Toul Hane	Format(a)	Bus Tool Astimutically
Denign Antry/Syn	dice >	+ starm?	Rate the last entered ally to synthesize the current design
Dimilation .	(hime>	- states	Run gate level smulation automatically after complation
liming Analysis	choire>	e cares	- i siusi the tool autometically after compliation
formal Verificat.	(TRIVE)	a contraction of the second	
Beard-Level	Tining	«None»	(H)
	Synhall	chione>	×
	Signal Integrity	<none></none>	8.
	Boundary Scan	«None»	2

Figure 7-6 New Project Wizard: EDA Tool Settings [page 4 of 5]

How Project Visard	
Summary [page 5 of 5]	
When you click Finish, the project will be created with the	following settings:
Project directory: Project name: Too-level design entity: Number of files added;	DL/mvftrst_ruosi mvftrst_ruosi mvftrst_ruosi 0
Further of user libraries added: Device assignments:	0
Panily name:	Cyclone IV E
Device: EDA toole	EPRCEAP17C6
Design entrajoyrithesis:	vitane (vitane)
Simulation:	diane> (diane>)
Timing analysis: Operating conditions:	(danes (danes)
vOCINF voltage:	1.27
Junction temperature range:	0-85 °C
	< Back heid > Pinets Cancel Helo

Figure 7-7 New Project Wizard: Summary [page 5 of 5]

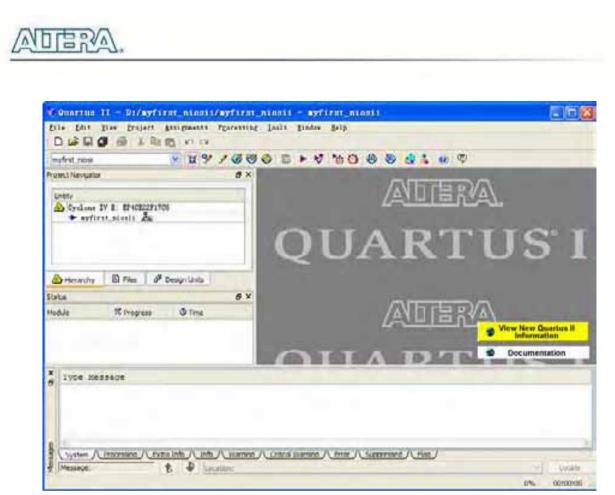


Figure 7-8 A New Complete Project

5. Select **Tools** > **SOPC Builder** to open SOPC Builder, the Altera system generation tool, as shown in **Figure 7-9**.

Ioula	Window Help	
Run Cn Lau	EDA Timing Analysis Tool nch EDA Simulation Library Compiler	•
1 Iim	eQuest Timing Analyzer	
Adv	izorz	٠
D Les	ign Partition Planner	
In- Log Inc Sic Dro M ITA	System Memory Content Editor ic Analyzer Interface Editor System Sources and Probes Editor malProbe Pins grammer G Chain Debugger msceiver Toolkit	
Qsy Tel Cus Qpt	C Builder m (Beta) Scripts tonige ions	
	Run Run Adv Chi Des Nat Sig Dro Sig Dro Ext Sig Dro Run Log Dro Run Log Dro Run Log Dro Run Run Run Adv Chi In Est Chi Sig Dro	Run EDA Simulation Tool Run EDA Timing Analyzis Tool

Figure 7-9 SOPC Builder Menu



na Create Readystes - Alt Fle Edi Module System Yew T		and super thirtagefore traces	eri Symmetri	enpi)	- 0 ×
System Contents System Generation	NA Deb				
Compenent Library	Target	Cisca Samaga			
Project Area component. Library # Avalon Verification Sule: # Bintiges and Adapters # Mentace Protocols # Mentace Protocols # Mentar Components # Mentar Components # Mentar Components # Portphorals # Int. # Portphorals # Int. # Portphorals # Avalen Components # Aval	Devide Family Cyclene IV E	Aane So	ate :	842	Add
	Use C. M. Create Mr	w System 🗧	Clock	Base	End
	Twyet HOL () V O V I This Specify Treases () Into Specify ()	INCL. a new system name. DK. Cancol	P+- Etters-		
	Ext Help	4 Parr Reat >	Generate		

Figure 7-10 Create New SOPC System [0]

6. Rename System Name as shown in **Figure 7-10** and **Figure 7-11**. Click **OK** and your will see a window as shown in **Figure 7-12**.

Greate New System	
System Name: DE0_NANO_SOPC	
Target HDIL:  Venilog VHDI.	
OK Concel	

Figure 7-11 Create New System [1]



# Alters SOPC Builder					
Ele Est Hotule System Yew I	oole lijele				
Eystem Contents System Germation					
Companient Liknary	Tarpe	Clock Sellarge			
Project A Description of the second	Device Family Cyclone N E	fiane cR_0	Source External	uk <u>r</u>  50.0	and and a
Interface Protocole     Ecgary Components     Memories and Memory Contro     Nortin Components     Provide and Memory Contro     Nortin Components     Provide and Memory Contro     Software Components     Pous     Provide and Memory     Yourgan     Video and Insole Processing     Yourgan	Use C. Muduk	Description		XxX Depr	End
Rev. to: Asc	4   10004   100   100   100   100		Address Meg.	10'3. Film Default	2
	Egit Help	4 Presi Next	Cenerale		

Figure 7-12 Create New System[2]

7. Click the **clk\_0** name in the Clock Settings table to rename **clk\_0** to **clk\_50**. Press **Enter** to complete the update, as shown in **Figure 7-13**.

S Altera SOPE Musider					
[is [dt Haduls System Yow ]	oola Halp				
System Contents System Ceneralise		-			
Companient Library	Target	Casca Semings			
Project → ☆ Now component Lithrary ⇒ Avaion Verification Suite ⇒ Or-Vares and Adapters ⇒ their face Protocols	Device Family Cyclone N E	Name	Source	NH2	Abl
		cak_50	External	50.0	(Demand
Legacy Components     Monories and Monory Contro     Marker Components     Pergherals     PLL	Use C. Module	Description	0	ock Base	End

Figure 7-13 Rename Clock Name

8. In the left hand-side Component Library tree, select **Library > Processors > Nios II Processor** and click the **Add...** button to open the Nios II component wizard, as shown in **Figure 7-14** and **Figure 7-15**.

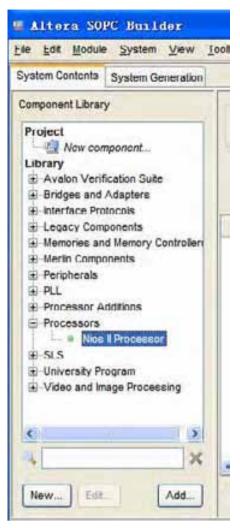
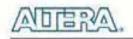


Figure 7-14 Add NIOS II Processor



Ni	05	II Processo	r		About Decementation
lacameter lettings			-		
	Cache	and Memory Interface	Advanced Features	MMU and MPU Settines JI AC Debug	Nodule Custom Instructions
Care Niap II					
Select a Nios II co	re:				
		ONios II/e	ONios II/s	○Nios II/f	~
NIOS II Seascor Cuida Famay Cyclone I foyclam 50.0 MHz cpuid: 0	ve	HISC 32-bit	HSC 32-58 Austraction Coulter Branch Prediction Hardware Multiply Bardware Divide	RSC 32-bit Instruction Conduct Branch Prediction Hardware Multiply Nantware Divite Darrel Shifter Usta Coche Dynamis, Branch Prediction	
Performance at 50.			Up to 32 UMPS	Up to 57 DWPS	
Logic Usage	-	600-700 LEs	1200-1400 LEs	1400-1800 LEs	9
fantware Mutiply	Embed	Ked Multipliers	Mandware Divide		
React Vector:	Mome	0	Giffet og	0	
Exception Vector	Momor	v.	Offset 0x2	0	
include MMU			and the second second		
Fast TLB Miss Exco			stem that explicitly supports an M	Offset	
	2011	and the second s			
Include MPU					
Warning: Heast	vector	and Exception vector ca	innet be set until memory devices	are connected to the floor 8 processor	
	_				

Figure 7-15 Nios II Processor

9. Click **Finish** to return to main window as shown in **Figure 7-16**.



Component Library     Target     Cock Settings       Project I Water component Library     Rame     Source     Mit     And       In Avison Vortication Syste is inveges and Addents     Resonance     Mit     And       Is before Protocals     Before Comm     Module     Description     Source     Mit       Is before Protocals     Before Comm     Module     Description     Cock Settings       Is before Protocals     Before Comm     Module     Description     Cock Settings       Is before Protocals     Before Comm     Module     Description     Cock Settings       Is before Protocals     Before Comm     Module     Description     Cock Settings       Is before Protocals     Before Comm     Module     Description     Cock Settings       Is Provide and Protocals     Before Comm     Module     Description     Cock Settings       Is Provide and Processorie     Is Spall     Before Common Setting     Arabin Hemory Mapped Statese     Is State       Is University Program     Is Spall     Is and Tampe Processorie     Is State     East       Is University Program     Is It is an It	System Centerts System Generation						
Project     Name     Source     Name       Water component     Ubrary       Water component     Water component       Water component     Water component <th>Contraction of the second seco</th> <th>Target</th> <th>Clock Settings</th> <th></th> <th></th> <th></th> <th></th>	Contraction of the second seco	Target	Clock Settings				
Image: Construction     Module     Description     Ouck     Base     End       Image: Marine Components     Image: Marine Components     Image: Componens     Image: Componens     Image: Comp	Library E Aven Vortification Sylle	And the second participation of the second					
	Ecquery Conjunction     Memories and Memory Controllers     Merin Components     Proybecols     Provessor Additions     Processors	C C Spu 0 samucteo_ma data_master	Nee & Praces Avalant stend Avalant Veno	ry Mapped Master	[ck] cik_sti [ck]	180	
New, Lot Add Degree (co. X A Y X Address Map [Bers] Filer Default	<ul> <li>University Program</li> <li>Video and Image Processing</li> </ul>						
	E University Program     Video and Image Processing	×					

Figure 7-16 Add Nios II CPU completely

10. Select the **cpu\_0** component and right-click then select rename, after this, you can update **cpu\_0** to **cpu**, as shown in **Figure 7-17** and **Figure 7-18**.

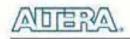


and the second se		_							
System Contents System Gaveration									
Component Library	Target		Can's Delings						
Project	Device Family Cyclone N E	36	Name		Source		MHz.	ADD	
Library Library La Avaion Ventication Suite Le bridges and Adapters			all_60 Externo		Enternal	60.0		(Hermonic )	
Initiation Protocols     Edgacy Components	Use Conn. Module		Descri	0506		Clock	Sales	End	
Menores and Menory Controleri     Marin Components	COLUMN COLU	Canne	10000			icki .			
e Processor Additions e Processor Additions Processore a St.5 e St.5 e University Program e Video and Image Processing		Devel Show Defeul All Clocks Avalue Avalue	-444		Mapperf Master Mapped Serve	icei Icei	= 0x0000	794 0 DEGG ONDOUGOITI	
1. E		£44	:01	r1+2	-				
× ×	0	Regard		1118					
New- Les Add.	Rageve gat	Region Details		,	Addresse Map	[Reva.]	Film: Default		
To Du. upu: 0. No react vector has to	per scentieri he ins CPU. Pere	Drew	Arbitration		Units Instant				
To Do: opu_0. No exception vector h	as been specified for this OPU.	Lock have schemen clarged.		1 A T A	stvo the issue				
To be: cpu_s. No exception vector has been specified for this CPU. S. Warning: cpu_8: Revel vector and Exception vector calmit he set all the set of the set of th		Expansion		-	ly line ligne il proceetant				

Figure 7-17 Rename the CPU (1)

stern Particular I and a start of	Doute Here & Histo				
ratem Contents   System Generation  omponent Library	Targol	Chick Settings	_		
Project Mer component. Library In Avision Venification Suite In Bridges and Adapters	Device Family Cyclone IV E		arce	MM2 MOIO	Add
Interface Protocols     Legacy Components	Use Conn. Modue	Description	Cent	Dage	Ent
Monorise and Menory Controller     Merkin Components     Portion als     PLL     Processor Additions     Pocessors     Pocesors     Pocessors     Pocessors     Pocessors     Poces	P B Cou instruction_max data_maxter jag_deteg_mo	Avainn Memory Mapped B	laster (ch)	TRO UKCOOQUBU	n a
	C. C	IT WATER AND	and a grant to be seen as	Filer Defaul	
New Lat. Adt	Remove Edt = 4	T I Address M	ao. Fiters		

Figure 7-18 Rename the CPU (2)



11. Add a second component by selecting Library > Interface Protocols > Serial > JTAG UART and clicking the Add... button, as shown in Figure 7-19 and Figure 7-20.

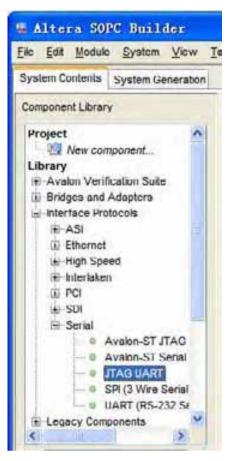


Figure 7-19 Add the JTAG UART component



I JTAC UART -	jtag_uart_0	٥
JTAG	UART	Documentation
* Block Diagram		
roc	ck clk clk reset avalon_itag_slave	irq interrupt
Write FIFO (Data fre	om Avalon to JTAG)	
Buffer depth (bytes): IRQ threshold:	8	
	egisters instead of memory t	blocks
Read FIFO (Data tro	m JIAG to Avaion)	
Butter depth (bytes):		
IRQ threshold:	8	
Construct using r	egisters instead of memory t	blocks
Simulated input ch	aracter stream	
Contents:		
Prepare interactive	: windows	
Options.	INTERACTIVE_ASCI_OUTP	UT 💌
		Cancel Finish

Figure 7-20 JTAG UART's add wizard

12. We are going to use the default settings for this component, so click **Finish** to close the wizard and return to the window as shown in **Figure 7-21**.



ystem Contentis System Generation						_
Component Library	(Terus) :	Calva Sellinge				
Project  Ver component Library Avaion Ventioation Eute E Dicipes and Adapters	Device Family Cyclone IV E	Name FR. 50				Ad3
i intertane livetanesia III A⊈I	Use Conn. Module	Description		Clock	Base	End
Ethernet     High Speed     High Speed     High Mon     PCI     B SDI     Serial     Avainn-ST JLAO     Avainn-ST JLAO     Avainn-ST Serial     SP(3) Ware Serial     UVIT (NS-332 Se     Uvit (NS-332 Se     Uvit (NS-332 Se     V))     SP(3) Ware Serial     V)     V)	Image: Construction of the second s	Avaion Monor Avaion Monor JTAG UART	y Maspeni Master y Mapped Dasser y Mapped Sileve y Mapped Sileve	(48,50 (48,50 (48) (48) (48) (48,50	190 2 8x60050000 6 0x00000000	0 0.55556.fr 0#000000
New Los Leon	Kegove [ct ] .		Address Mag	Laura es	er Defeit	2
To the epur tin reset vector has bee To Do, opur Ne exception vector has	in specified for this CRU. Please parameter a boon sociolited for this CRU. Please para option vector cannot be set until memory	entre the CPU in resolve the	tis issue No this issue			

Figure 7-21 JTAG UART

13. Select the **jtag\_uart\_0** component and rename it to **jtag\_uart** as shown in **Figure 7-22**.



System Contentis System Generation						_
Component Library	Target	Citize Settings				
Project	Device Family  Cyclone IV E	Name Coci So	Source External	-	0 M1	Add
i intertace livetacela	Use Conn. Module	Description	-	Clock	Base	End
<ul> <li>Ethernel</li> <li>High Speed</li> <li>Interface</li> <li>FCI</li> <li>SDI</li> </ul>	E opv Instruction, mai data_master Jag_debug_me	stor Avalor Mon Avalor Mon		[v8] c8, 59 [c8] [c8]	110 0 • • • • • • • • • •	
Avaine.51 Server     TAO BART     SPI (3 Way Seriel						
+ UART (RE-232 5)						10
+ UA/IT (RE-232 Ex in: Legacy Components M	Kangova (Lat ) X a		( Аскалала Мар. )	Liters +as	e Delaut	2

Figure 7-22 Rename JTAG UART

15. Add the Library > Memories and Memory Controllers > On-Chip > On-Chip Memory (RAM or ROM) component to system, as shown in Figure 7-23 and Figure 7-24.



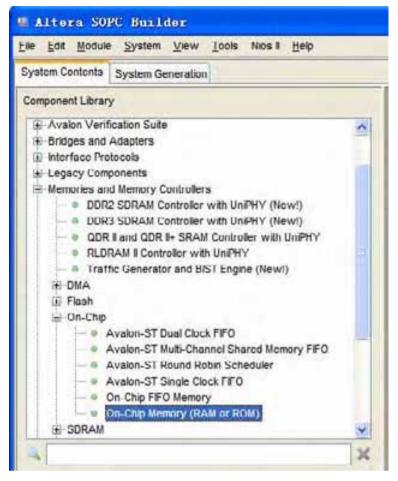


Figure 7-23 Add On-Chip Memory

ADERA.
--------

	RAL or ROL) - onchip Nemory (RAM or ROM nchip_memory2	
* Block Diagram		4
	clock clict ovalon reset1	0
* Memory type		
Туре:	RAM (Writable)	
Dual port access		
Read During Write Mode:	DONT CARE Y	
Block type:	Auto 🖌	
* Size		
Data width:	32 💌	
Total memory size:	4096	bytes
Minimize memory block	usage (may impact fisial)	
* Read latency		
Slave a1 Latency:	1 💌	
Slave a2 Latency:	1.4	
* Memory initialization		
Initialize memory conter	t	
Enable con default initia	Eration file	
		Cancel Finish

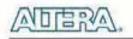
Figure 7-24 On-Chip Memory Box

16. Modify Total memory size setting to **26000** as shown in **Figure 7-25**. Click **Finish** to return to the window as in **Figure 7-26**.

* Block Diagram	onchip_memory2	Root	umentation
	clock clk1 ovalon clk1 reset reset1	ry2_0	
* Memory type			1
Туре:	RAM (Writable)		
Dual port access			
Read During Write Mode:	DONT CARE		
Block type:	Auto 😒		
* Size			
Data width:	32 💌		
Total memory size:	26000	bytes	
Misinice memory blog	ck sineader (may implect finasic)		
* Read latency			
Slave a1 Latency:	1 ¥		
Slave s2 Latency:	1.0		
Memory initialization			
Initialize memory cont	lent		
	Halization fla		

Figure 7-25 Update Total memory size

70)



palant Darferter System Oregenites					
Concernment Library		Trees	Char Service		
in Acatori Vanthatina Suite     in Strappin and Antephone     in ther their Protection     in ther their Protection     in Commonly Antephone     in Ammonia and Ministry Comparison     in Difference and Directory Comparison     in Difference and Directory		Chevice Family Cystone 6/2	Rate Starts	911	
DORO SDRAM Caretyler web (IniPHY IN		the cont. tradue	Descouter	1000	3000
SDR Fand COR Is SRAM Controllar sets     DLPANE Is Controllar sets     DLPANE Controllar sets     DLPANE     DLPANE     DLPAN     Frank     Frank     Control	CALCON	E E ser Refraction real data_moder Jag_ Refraction real	Avaies Densry Barrow Master		- 042
Andres ST Mar Cean TFD     Andres ST MyG Channel Parret Inc     Andres ST MyG Channel Parret Inc     Andres ST MyG Channel Parret Inc     Andres ST MyG Channel Parret     Andres ST MyG Channel Parret     Social Parret     Social Parret     Social Parret     Social Parret		E C suite server(2)		08_50  181   48_50	1. 540
	18	0	and the second se		
See	A.M.	[fegere] 544. [ 36 ]		[ [ Area ] 70	ec Defeut
Error operatestmattion measure overhap memory (A) Tron operates measure country memory (A) To De operate overhap overhap been apouted for To De operate to maniplen vertice test been specified	n carnet be al United	o can a private the acceptable i analysical the CRI to resolve the educ-			

Figure 7-26 Add On-Chip memory

17. Rename onchip\_memory2\_0 to onchip\_memory2 as shown in Figure 7-27.

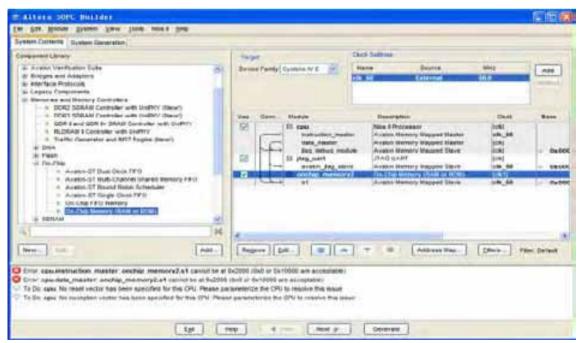
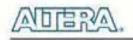


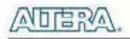
Figure 7-27 Rename On-Chip memory

18. Right click on the **cpu** component table and select **Edit...** from the list. Update the Reset Vector and Exception Vector as shown in **Figure 7-28**. Then, click **Finish** to return to the window as shown **Figure 7-29**.



E Rios II Pre	00098	ior – gpu				8
Ni Ni	ios 1	II Processo			About Dares	entation
Parameter Settines			A CONTRACTOR OF			
	Calm	end Menury Interfaces	> Advanted Produces > P	Milaul MPU Sellings > 11	Ali Debay Module 🔰 Custom 1	minuture
Core Mos II Select a Nos II co	нис					
		ONIOS II/e	ONIOS IL/S	INIOS II/T		-
NIOS II Delector Guide Family: Cyclone I ( <sub>System</sub> , No.0 MH) cpuid, 0	NF	185C 32-60	REC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide	RISC 32-Jul Instruction Cache Dronch Prediction Hartware Islattpy Hardware Oriele Resuel Statter Data Cache Dynamic Dranch Prediction	n	
Porformance at 50.		Contract Services of the	Up to 32 DMPS	Up to 57 OMPS	<b>1</b> 11	
Logic Usage	1	600 700 LEB	1200-1400 LEN	1400-1800 LEE	-1	2
Hardware Multiply:	Embed	ided Multiplers	Hardware Davide			
Roast Vooter:		W unchip memory2	Offact nen	0x000	02000	
Exception Vector:	Memor	Y ancho_menory2	Qttset: 0x20	0x0000	2020	
include UNU						
Only include the MM Fast TLB Mile Exce		CARLS IN A REAL PROPERTY AND ADDRESS	tem that explicitly supports an MMU	Offset		
Include MPV						
						2000
					Cancel Link Leat	Ernes

Figure 7-28 Update CPU settings



engeneti Likrary + Avaon verification suite + Ondee and Adapters	Tartet Deute Family Cycline IVE 4	Click Setings		
	Device Family Curines I/F			
e: Hertace Protocels & Logscy Components # Menories and Menory Controllers	Controlling Section 8. (2)	Name Source	NPTZ SALA	Add
OR2 SORAM Custofier with UniPHY (New)     DOR3 SORAM Custofier with UniPHY (New)     OCH Fault COM In SHAW Custofier with UniPHY	USA CAAR MARA	Description	Cave	Hane
PLDRAM # Controller with UniPHY     Traffic Controller with UniPHY     Traffic Controller and BIST Engine (Nerwi)     DIVA     Filesh     Con Che     Avalues ST Dual Clock FFO     Avalues ST Busic Clock FFO     Avalues ST Busic Submitute     Avalues ST Busic Submitute     Avalues ST Busic Submitute     Avalues ST Busic Clock FFO     Con Che FFO Nemory     Con Cloc FFO     Con Cloc F	In Colu     Columnation     Columnation	Availor Namory Happed Natter dule Availor Namory Happed Dave JTAO UART we Availor Memory Happed Dave	(ck) (ck) ck, 50	- 0+0 - 010 - 0x0
Sew (c) Adt	Regione Est a	Assess Map	Emers P	der Detaut

Figure 7-29 Updated CPU settings

19. Add the Library > Peripherals > Microcontroller Peripherals >PIO (Parallel I/O) component to the system, as shown in Figure 7-30 and Figure 7-31.

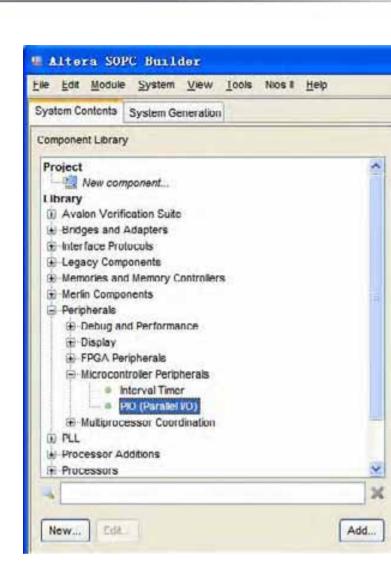


Figure 7-30 Add PIO

to 0)

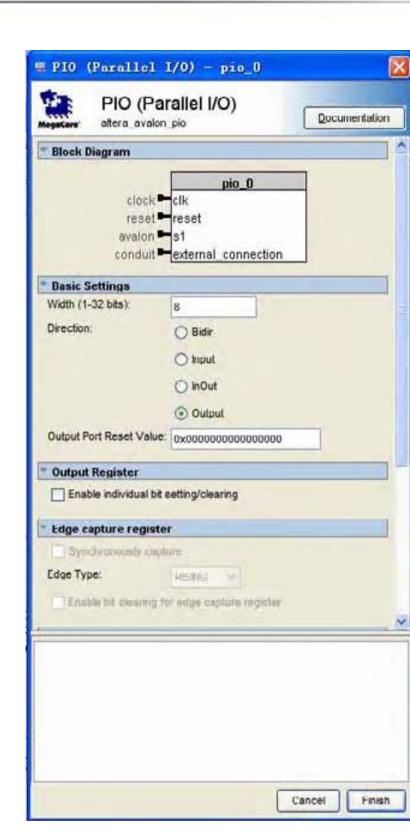


Figure 7-31 Add PIO

20. Click **Finish** to use the default settings for this component. This closes the PIO wizard and returns to the window shown in **Figure 7-32**.



deni Cuntenila System Generalist				
ngunani Library	Thight	Clock Setting		
Report Component	Device Family Cyclame IV E	Name Dource	MHg KLA	Adt
E-Interface Protocolls 2 Legary Components 2 Materials and Materia Controllars	Vec - Deen. Hodule	Description	Devit	
I Merile Cangasenta Perspheras Perspheras Colog and Performance Display PERCA Resplayeda Displayed Person other Perspheras Displayed Town Person Per	Image: State of the s	Availab Meessry Mapped Master dule Availab Meessry Mapped Devo ITAG MART Availab Meesery Mapped Devo	10,59 (44) (44) (44) (44) (44) (44) (44) (44	= 000 = 8x0 = 8x0
ten tel	Resourt Edit. II 4	Address Mag	Etters Fa	her Default

Figure 7-32 PIO

21. Rename **pio\_0** to **pio\_led** as shown in **Figure 7-33**.

Annerska and Memory Eachtrakers     Avalam Manary Eachtrakers     Avalam Manary Eachtrakers     Avalam Manary Eachtrakers     Avalam Manary Eachtraker     Avalam Manary Manary Eachtraker     Avalam Manary Manary Eachtraker     Avalam Manary Manary Manary Manary     Avalam Manary Manary Manary     Avalam     Avalam Manary     Avalam     Avalam Manary     Avalam	E	Ē							a SADT Rei Lder	
Categorient Library     Target     Deck Settinge       Project     Device Family     Cyclone N/E     Name       Mean composent     Device Family     Cyclone N/E     Name       Manacisa and Adapters     Device Family     Cyclone N/E     Name       Introduces and Memory Controllers     Device Family     Cyclone N/E     Name       Manacisa and Memory Controllers     Device Family     Cyclone N/E     Name       Manacisa and Memory Controllers     Device Family     Device Family     Cyclone N/E       Wee     Comm     Module     Descrution     Cock       Parguments     Parguments     Avain Memory Mapped Master     Chill       Wei     Cock     Parguments     Avain Memory Mapped Stave     Chill       Wei     Stand Memory Mapped Stave </th <th>-</th> <th></th> <th></th> <th></th> <th></th> <th>_</th> <th>-</th> <th>eri Don</th> <th></th> <th>Conception of the local division of the loca</th>	-					_	-	eri Don		Conception of the local division of the loca
Avalant Verification Studie     Avalant Verification     Avalan					leck Detringe		Tarpe		The second second second second	
Use     Conn.     Module     Destrution       Water Score     Conn.     Module     Destrution       Pergmense     E Conn.     Module     Destrution       In Destrution     Pergmense     Pergmense     Conn.     Module       In More Name     Pergmense     Pergmense     Conn.     Module       In More Name	A31		1000			Hy Cyckne N E	Device	*	on Verification Sixte	Library
Works Cheposents     Frequences			Clock	lan .	Descrut	nn. Module	Use		sox Protocole Cy Componenta	E Interface RE-Legacy C
Processor Additions	Dx 000		ck, 59 [4] [4] [4] [4] [4] [4] [4] [4]	smory Mapped Master enery Mapped Uaster enery Mapped Sieve RT enery Mapped Sieve ferency (RAM or RDM) enery Masped Sieve	Avalum Ma Avalum Ma Avalum Ma JTAG UAJ Avalum Ma Ge-Dhip M Avalum Ma	<ul> <li>Fatracian, mask</li> <li>Seg_initian</li> <li>Seg_</li></ul>	E E		s Components mesos estog and Partermatice moley POA Peripherals Instructorder Pergherals = toter lineer = (contension) ungrocessor Contension reace Additions	Merin Co     Pergeners     Polymers
X O	10						0	×		-
New., 10. Add., Pageve Ltt., M. & W. M. Address Nap., [Revs., File: 1	test	File De	[Rers.]	Address Map.	1 1	En	Rege	Add	0.001	New_

Figure 7-33 Rename PIO



22. Select **System > Auto-Assign Base Addresses** as shown in **Figure 7-34**. Then, select **File > Refresh System**. After that you will find that there is no error in the message window as shown in **Figure 7-35**.

Cypton Cantorita				
	dh Annin Sam Addressa			
Contractional Libertaria	IO-Assign Higs ant Arasim-ST Adapters	Target	CANA DetRoga	
Project 30 here contribution 1 brany Analise Verification 3 Orders and Adaptor branches, Networks		<ul> <li>Denice Family Cycline M.E. M.</li> </ul>	Nana Soute DS_50 Kxtertaa	Mriz Add
# Legacy Components		User Cores Modum	Owneld & down	Clock Rand
Honoras and litence Morio Componente Protoletata Usaper Proto Anepteran Baccomotive A Proto Anepteran Proto Anepteran Protocesor G Pt.1 Processor Adatame Processor Adatame Processor Adatame	Taner Taner Bartusta Sourcestanation	Image: state	Avalor Menory Mapped Haster due Avalon Memory slapped Save STAC UAUT and Avalor Menory Mapped Save	[14] (14]
-		X		
Alexand French	4	AL. Report \$18	w 🕱 🗛 Address Map.	ERers. Filter Default

Figure 7-34 Auto-Assign Base Addresses

Vatern Contents Rystein Generation					
Component Library	Tariat	Clean Detterge			
Project  Aver component Library A Avelon Verification Sulls Distyre and Adapters	Device Family Cyclone N E 9	Name clk_50	Sours Faternal	10159  50.0	Add
i interface Protocols ii Lagety Components	Line Const. Monlum	Description	un	Clenck	Base .
DORC SUBKAN Controller with Un     OCH1 SOFLAN Controller with Un     OCH1 Soft Control Centroller with Unit's     REDRAW & Declaration with Unit's     Traffic Generative and RIST Frage     DMA     Flash     House     Softam     Machiner with Unit's	Image: static	Availor tee dule Availor te JTAG UAR Availor Me Availor Me Availor Me PQ (Paralle PQ (Paralle	mory Mappert Master mory Mapped Master mory Mapped Slave Trory Mapped Slave embry (RAM or ROM) mory Mapped Slave	10.82 edia_60 [0.82 [4.8] cris_60 [ris1] cris_60 [ris1] cris_90 [153] cris_50	159 Dx 00016800 - 0x 00006000 - 0x 00006000
Nerw (III. Add. Warning: cour Custom Instruction components on Warning: cour Dealong the assign CPUE control		e.:	Address Map		er Default De furmed on with de

Figure 7-35 No errors or warnings



23. Click the Generate button, which will pop up a window, as shown in **Figure 7-36**. Click Save, which bring up the window in **Figure 7-37**. Input the name, **DE0\_NANO\_SOPC**, and click the save button. The compilation will automatically start. If there are no errors in the generation, the window will show a message of success, as shown in **Figure 7-38**.

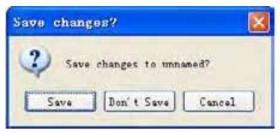


Figure 7-36 Generate SOPC

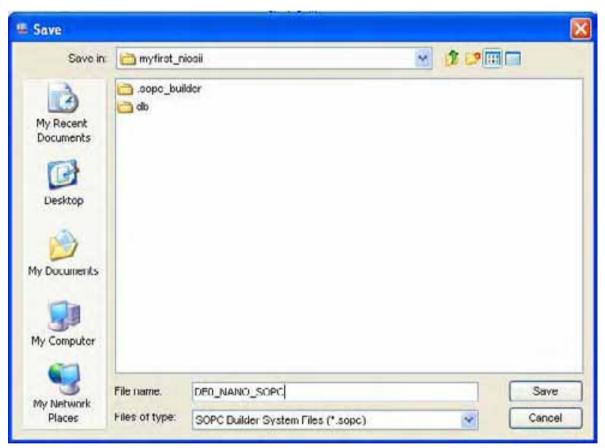


Figure 7-37 Generate SOPC

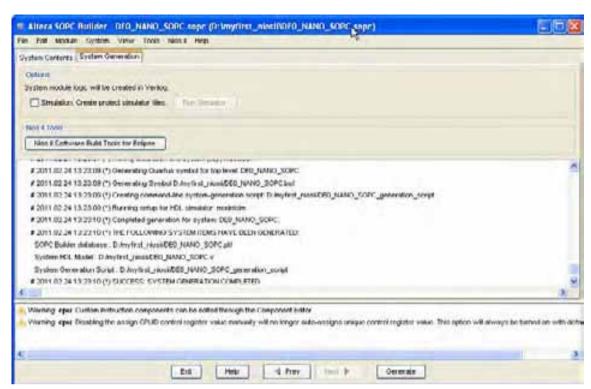


Figure 7-38 SOPC Builder generation successful

24. Click Exit to exit the SOPC Builder and return to the window as shown in Figure 7-39.

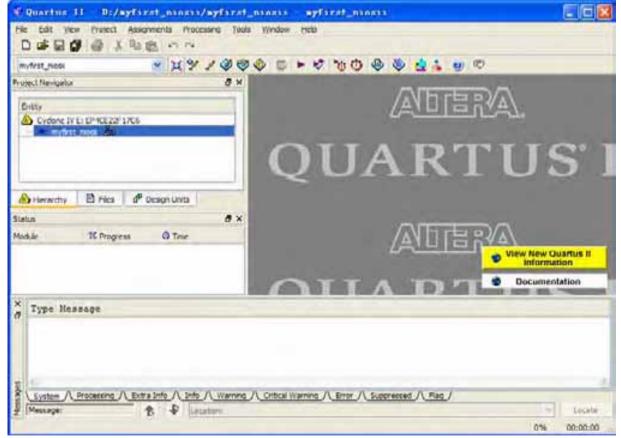
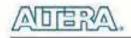


Figure 7-39 Return to Quartus II after exiting SOPC Builder



25. Create a new Verilog HDL file, by selecting **File > New**, **Verilog HDL File** and click **OK**, as shown in **Figure 7-40** and **Figure 7-41**.

비는	Fort Tiem Flobe	ct Assignments	Proces
D	New	CEIHN	4
C)	Open	CH+0	30
	Qax-	(1)(1)(1)	1
2	New Project Wizard		
C.	Open Project	Ct+I+J	
	Seve Project		
	Close Project		
Q	<u>Eave</u>	CT145	
	Seve Series		
ø	Save All	Ctrl+Shift+5	_
	Ele Properties		T
	Create / Update		P
	Faiscil		
	Convert Programming i	Files	-
1	Page Seign		
B	Print Preview		
8	Part	Celép	
	Recent Files	•	
	Recent Projects	•	5
	Exit	Alt (E1	

Figure 7-40 New Verilog file

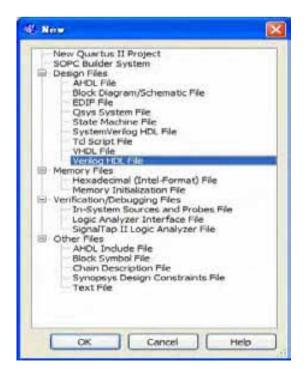
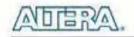


Figure 7-41 New Verilog File



33. Figure 7-42 show a blank Verilog file.

wyfest_ronsi eject Na-igieter	× x × .	/ 🖓 🧐 # x	0	D) P	10			-		191 I	e)				
Cydana IV Ei BPACE20P Tryfrest recea dda Prenarchy M Pies atus uskie TE Progra	d <sup>®</sup> Design Units	••		<b>Ph</b> 15		te te	 3	. 16	. **		172	2) x	I whi	T	 5
		4	e												*
Type Nessage															

Figure 7-42 A blank verilog file

34. Type the following Verilog into the blank file, as shown in **Figure 7-43**. The module **DE0\_NANO\_SOPC** is the system created by SOPC Builder and its Verilog can be found in the **DE0\_NANO\_SOPC.v** file, as shown in





Figure 7-44 and Figure 7-45.

```
module myfirst_niosii
(
    CLOCK_50,
    LED
);
input
               CLOCK_50;
output [7:0]
             LED;
DE0_NANO_SOPC DE0_NANO_SOPC_inst
    (
      .clk_50
                                  (CLOCK_50),
      .out_port_from_the_pio_led (LED),
      .reset_n
                                  (1'b1)
    );
```

### endmodule

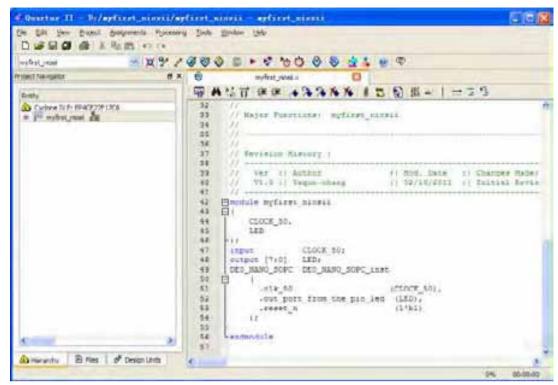


Figure 7-43 Input verilog Text



Figure 7-44 Open DE0\_NANO\_SOPC.v

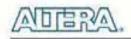
and the second se		irst, statti – apfirzt, statti	
Dis Edd year Dream	And the state of t	Dawn Warden Reh	
	a the second		
mphat_oldi		9 0 0 F 4 0 0 0 0 2 1 0 0	
right Neilgeter	# X	🐮 myfret_week > 🖸 🕼 DED_WWO_SOFC > 🚺	
Entry		國具作品 化化学学学学学学生 医胆酸素 一年二日	
Contore IN Et FF-KE225:     * myffrei ynwit All		2007 2008 Growile DES_WANG_SOPC / 2009 sik_00, 2001 seat_s, 2002 // the_pis_les 2003 // the_pis_les 2004 / 2004 / 2004 / 2005 / 2004 / 2005 / 2005 / 2006 / 2006 / 2006 / 2007 / 2008 // the_pis_les 2008 / 2009 / 2009 / 2009 / 2009 / 2009 // 11 global Algoals, 2009 // the_pis_les 2000 // the_pis_les	
		2100 input sees_al	
12 Martin - The second	and the second s	2101 2102 wile ulk 30 recet 40	
Aneathy Bries		and the second s	3
Type Ressaue			
Suttern A microsoft /	Lanson Auto Au	aries A creat marine A true A subjessed A rise J	5. 00 m dd

Figure 7-45 DE0\_NANO\_SOPC module

35. Save the newly created Verilog file as **myfirst\_niosii.v**, as shown in **Figure 7-46**.

Save As					2 🔀
Save in: My Recent Documents Desktop My Documents My Computers My Network	ff cpu_itag_de	r SOPC_sm bug_module_syscik.v bug_module_tick.v bug module wrapper.v il.v s_bench.v nch.v sOPC_v SOPC_inst.v			
Placev	l de geme:	mydest_mose.v	 -	1 [	Save
	Save as type:	Vening HDI Files (* v * v Add file to cuttent pro		1	Cancel

Figure 7-46 Save the Verilog file



36. Compile the project, by selecting **Processing** > **Start Compilation**, as shown in **Figure 7-47**. **Figure 7-48** shows the compilation process.



Figure 7-47 Start Compilation

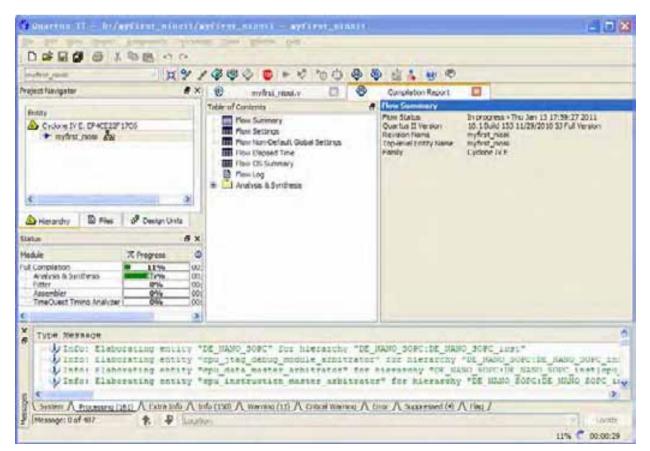
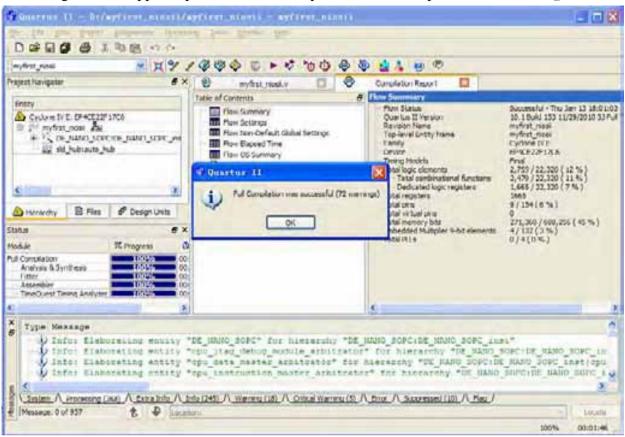


Figure 7-48 Execute Compile





37. A dialog box will appear upon successful completion of the compile, as shown in Figure 7-49.

Figure 7-49 Compile project completely

38. Now, we will assign the inputs and outputs of the circuit to specific pins. Select **Assignments** > **Pin Planner** from the menubar, as shown in **Figure 7-50**. The pin planner is shown in **Figure 7-51**.

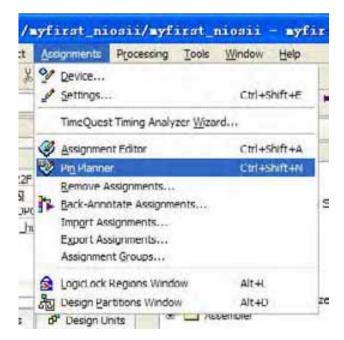
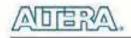


Figure 7-50 Pins menu



Node Name	Dreston	Lucaban	I/O Bank	VREF Gruup	I/O Standard	Reserved
EP 0100K_50	Input				2.5 V (default)	
CEP LED[7]	Output				2.5 V (default)	
1 LED[6]	Output				2.5 V (default)	
1 LCD[5]	Output				2.5 V (defoult)	
(4) 180[4]	Cutput				7.5 V (/setault)	
1 LED[3]	Output				2.5 V (default)	
(H0[7]	Output				2.5 V (default)	
@ LLD[1]	Output				2.5 V (default)	
@ LED[0]	Output				2.5 V (default)	
< <new node="">&gt;</new>	and the second sec					

#### Figure 7-51 Blank Pins

39. Input Location values as shown in Figure 7-52.

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
LOCK_50	Input	PIN_R8	3	83_NO	2.5 V (default)	
@ LED[7]	Output	PIN_L3	2	82_N0	2.5 V (default)	
D LED[6]	Output	PIN_61	1	51_10	2.5 V (default)	
0 LCD[5]	Output	PIN_CO	1	D1_N0	2.5 V (default)	
DED[4]	Output	PIN_D1	1	B1_N0	2.5 V (default)	
CV LED[3]	Output	PIN_A11	7	87_10	2.5 V (default)	
CP LED(2)	Output	PIN_B13	1	87_N0	2.5 V (default)	
UED[1]	Output.	PIN_A13	1	87_N0	2.5 V (default)	
() (CD[0]	Output	PDI_A15	7	07_10	2.5 V (default)	
< <new node="">&gt;</new>						

Figure 7-52 Set Pins

40. Close the pin planner and recompile the project.

# 7.3 Download the Hardware Design

This section describes how to download the configuration file to the board.

Download the FPGA configuration file (i.e. the SRAM Object File (.sof) that contains the NIOS II based system) to the board by performing the following steps:

- 1. Connect the board to the host computer via the USB download cable.
- 2. Start the **NIOS II IDE**.
- 3. After the welcome page appears, click Workbench.
- 4. Select Tools > Quartus II Programmer.
- 5. Click Auto Detect. The device on your development board should be detected automatically.
- 6. Click the top row to highlight it.

### 7. Click Change File.

- 8. Browse to the myfirst\_niosii project directory.
- 9. Select the programming file (myfirst\_niosii.sof).
- 10. Click **OK**.
- 11. Click **Hardware Setup** in the top, left comer of the Quartus II programmer window. The Hardware Setup dialog box appears.

12. Select USB-Blaster from the currently selected hardware drop-down list box, as shown in **Figure 7-53**.

Note: If the appropriate download cable does not appear in the list, you must first install drivers for the cable. Refer to Quartus II Help for information on how to install the driver.

Hardware Setup			
Hardware Settings JTAG S	ettings		
Select a programming hardware hardware setup applies only to			s. This programming
Currently selected hardware: Available hardware items	USB-Blaster [USB-0] No Hardware USB-Blaster [USB-0]		M
Hardware	Server	Port	Add Hardware
USB-Blaster	Local	USB-0	Remove Hardware
			Close

#### Figure 7-53 Hardware Setup Window

- 13. Click Close.
- 14. Make sure the **Program/Configure** option for the programming file (see **Figure 7-54** for an example).
- 15. Click Start.



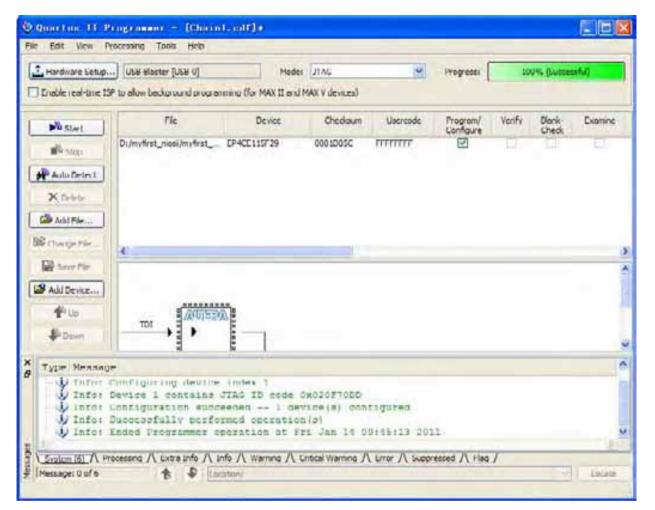


Figure 7-54 Quartus II Programmer

The Progress meter sweeps to 100% after the configuration finished. When configuration is complete, the FPGA is configured with the Nios II system, but it does not yet have a C program in memory to execute.

The Nios II IDE build flow is an easy-to-use graphical user interface (GUI) that automates build and makefile management. The Nios II IDE integrates a text editor, debugger, the Nios II flash programmer, the Quartus II Programmer, and the Nios II C-to-Hardware (C2H) compiler GUI. The included example software application templates make it easy for new software programmers to get started quickly. In this section you will use the Nios II IDE to compile a simple C language example software program to run on the Nios II system on your development board. You will create a new software project, build it, and run it on the target hardware. You will also edit the project, re-build it, and set up a debug session.

# 7.4 Create a hello\_world Example Project

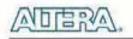
In this section you will create a new NIOS II C/C++ application project based on an installed example. To begin, perform the following steps in the NIOS II IDE:

1. Return to the NIOS II IDE.

Note: you can close the Quartus II Programmer or leave it open in the background if you want to reload the processor system onto your development board quickly.

- 2. Select **File** > **New** > **NIOS II C/C++ Application** to open the New Project Wizard.
- 3. In the New Project wizard, make sure the following things:
- a. Select the Hello World project template.
- b. Give the project a name. (**hello\_world\_0** is default name)

c. Select the target hardware system's PTF file that is located in the previously created hardware project directory, as shown in **Figure 7-55**.



#### New Project Nios II C/C++ Application Click Finish to create application with a default system library as D: \myfirst\_niosii\Software\hello\_world\_0 hello\_world\_0 Name . Specify Location Location: D: \myfirst\_niosii \Software Browse ... Select Target Hardware. SUPC Builder System FTF File: D'Amyfirst\_niosii\DE\_NAND\_SOPC ptf Browse .. CPU: 4 cpu Select Project Template Blank Project Description Board Diagnostics Frints 'Hello from Nios II' Count Binary Hello Freestanding Details. Hello MicroC/OS-II Hello World prints 'Hello from Nios II' to STBOUT Hello World Hello World Small This example runs with or without the MicroC/OS-II RTOS and requires an STBOHT device in your system's hardware Memory Test Memory Test Small Simple Socket Server Simple Socket Server (RGMII) For details, click Finish to create the project and refer Web Server to the readme txt file in the project directory Web Server (RGMII) 4 0 Hack Next > Finish Cancel

Figure 7-55 Nios II IDE New Project Wizard

5. Click **Finish**. The NIOS II IDE creates the **hello\_world\_0** project and returns to the NIOS II C/C++ project perspective, as shown in **Figure 7-56**.



Sine H Gill+ - hellspeeddur	- Rev 11 INF			
ble Bais Befeiger Berigete Segul De	opent Tools has Linder Help			
TI+		14410 + 6(+*)	0	5 Kass 12 2/044
non 11 0A* frojecta 11		Rello this dias 111 to the		***
	<ul> <li>duriges. It want wind</li> <li>Heving it group spates</li> <li>the seminy footplist</li> <li>using the statistic or</li> <li>ray a called Jorgers</li> </ul>	ni tais iretel applatation Recente design. 15. version ur this ineplate Inigrist for a given appla emplate.	I NICO AND PERSONNA A ATO IR -40 BRUTER RY DEFNICT , and an explanation of 5	
		225		10 17 11 15
	E Fraklant II Consells Properti 8 severa, O rurninga, O sadas			and the second second
	Jeseripties +	Basement Path	Laration	
ss	Vitable	Seart Steart 1 1		

Figure 7-56 Nios II IDE C++ Project Perspective for hello\_world\_0

When you create a new project, the NIOS II IDE creates two new projects in the NIOS II C/C++ Projects tab:

■ hello\_world\_0 is your C/C++ application project. This project contains the source and header files for your application.

■ hello\_world\_0\_syslib is a system library that encapsulates the details of the Nios II system hardware.

Note: When you build the system library for the first time the NIOS II IDE automatically generates files useful for software development, including:

• Installed IP device drivers, including SOPC component device drivers for the NIOS II hardware system

• Newlib C library: a richly featured C library for the NIOS II processor.

• NIOS II software packages which includes NIOS II hardware abstraction layer, Nichestack TCP/IP Network stack, NIOS II host file system, NIOS II read-only zip file system and Micrium's  $\mu$ C/OS-II realtime operating system (RTOS).

• **system.h:** a header file that encapsulates your hardware system.

• **alt\_sys\_init.c:** an initialization file that initializes the devices in the system.

• **Hello\_world\_0.elf:** an executable and linked format file for the application located in hello\_world\_0 folder under the Debug directory.

# 7.5 Build and Run the Program

In this section you will build and run the program.

To build the program, right-click the **hello\_world\_0** project in the Nios II C/C++ Projects tab and select **Build Project**. The **Build Project** dialog box appears and the IDE begins compiling the project. When compilation completes, a message 'Build complete' will appear in the Console tab, as shown in **Figure 7-57**.

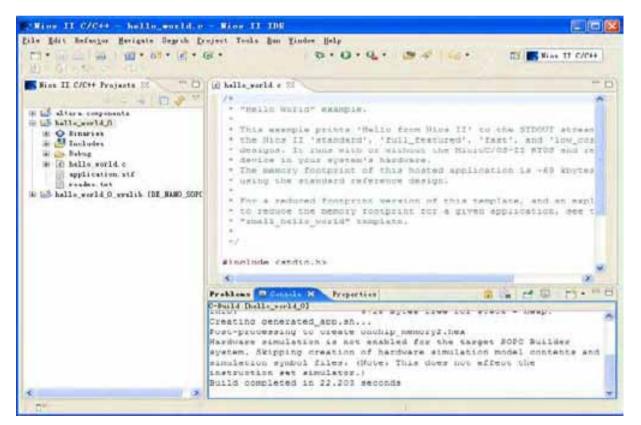


Figure 7-57 Nios II IDE hello\_world\_0 Build Completed

Note: If there appears in the console tab, an error, "region onchip\_memory2 is full(hello\_world\_0.elf section .text). Region needs to be XXX bytes larger.", please right-click hello\_world\_0, select System Library Properties menu, then pop a window. In the System Library Properties window, select Small C Library, then click OK to close the window. Rebuild the project.

# ADERA.

After a successful compilation, right-click the **hello\_world\_0** project, select **Run As > NIOS II Hardware**. The IDE will download the program to the target FPGA development board and begin execution. When the target hardware begins executing the program, the message '**Hello from Nios II!**' will appear in the NIOS II IDE Console tab, as shown in Figure 7-58 for an example.

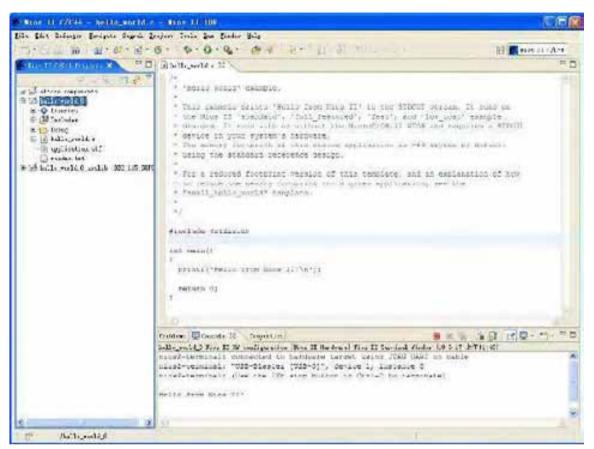


Figure 7-58 Hello\_World\_0 Program Output

Now you have created, compiled, and run your first software program based on NIOS II. And you can perform additional operations such as configuring the system properties, editing and re-building the application, and debugging the source code.

# 7.6 Edit and Re-Run the Program

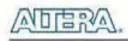
You can modify the **hello\_world.c** program file in the IDE, build it, and re-run the program to observe your changes, as it executes on the target board. In this section you will add code that will make the green LEDs, on the DEO-Nano board, blink.

Perform the following steps to modify and re-run the program:

1. In the hello\_world.c file, add the text shown in blue in the example below:

#include <stdio.h>

terasiC Terasic DE0-Nano User Manual



```
#include "system.h"
#include "altera_avalon_pio_regs.h"
int main()
{
printf("Hello from Nios II!\n");
int count = 0;
int delay;
while(1)
{
IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE, count & 0x01);
delay = 0;
while(delay < 2000000)
{
delay++;
}
count++;
}
return 0;
}
```

2. Save the project.

3. Recompile the project by right-clicking **hello\_world\_0** in the NIOS II C/C++ Projects tab and choosing **Run** > **Run As** > **Nios II Hardware**.

- Note: You do not need to build the project manually; the Nios II IDE automatically re-builds the program before downloading it to the FPGA.
- 4. Orient your development board so that you can observe LEDs blinking.

# 7.7 Why the LED Blinks

The Nios II system description header file, **system.h**, contains the software definitions, name, locations, base addresses, and settings for all of the components in the Nios II hardware system. The **system.h** file is located in the in the **hello\_world\_0\_syslib\Debug\system\_description** directory, and is shown in Figure 7-59.

Nior II 0/014 - systemsh - Nio	New York Constraints of the Cons			إلحا تع
11. Edit Befactor Herigate Degrah Dr. 13. An	and the second	n	Hi oi	II C/C++
Nies II C/C++ P. II CO C ha	lis world e 📓 swaten h 🐹			. m (
Debug In Constraints of the second	pio_led configuration we fine FIO_LED_HAME "/dmv/pio_lmd" efine FIO_LED_TYPE "sitexa avalon_pia" wfine FIO_LED_BASE 0x00011000 efine FIO_LED_BASE 0x00011000 efine FIO_LED_DG_TREE BENCH WIRING 0 efine FIO_LED_DATANISM_VALUE 0 efine FIO_LED_DATANISM_VALUE 0 efine FIO_LED_DATANISM_VALUE 0 efine FIO_LED_CAFTURE 0 efine FIO_LED_CAFTURE 0 efine FIO_LED_EDGE_TYPE "NONE" efine FIO_LED_BIT_CLEARING_EDGE_REGISTER 0 efine FIO_LED_BIT_MODIFYING_OUTFUT_REGISTER 0 efine FIO_LED_BIT_MODIFYING_OUTFUT_REGISTER 0 efine FIO_LED_FREQ \$0000000 efine ALT_NODULE_CLASS_PI0_led sitess_evalor_FAO			
erto d Probl	tees County IC. Properties	Inter of	1년 137	
Lon makefile	4 [hall+_s+r14_0_sys1ib]		Post. and	
The second secon	ting generated.x			
	d completed in 199.609 seconds			
R	TO THE REAL PROPERTY AND AND ADDRESS.			

Figure 7-59 The system.h file

If you look in the **system.h** file for the Nios II project example used in this tutorial, you will notice the **pio\_led** function. This function controls the LEDs. The Nios II processor controls the PIO ports (and thereby the LEDs) by reading and writing to the register map. For the PIO, there are four registers: **data**, **direction**, **interruptmask**, **and edgecapture**. To turn the LED on and off, the application writes to the PIO's data register.

The PIO core has an associated software file **altera\_avalon\_pio\_regs.h**. This file defines the core's register map, providing symbolic constants to access the low-level hardware. The **altera\_avalon\_pio\_regs.h** file is located in the directory, **altera\10.1\ip\sopc\_builder\_ip\altera\_avalon\_pio**.

When you include the **altera\_avalon\_pio\_regs.h** file, several useful functions that manipulate the PIO core registers are available to your program. In particular, the macro

IOWR\_ALTERA\_AVALON\_PIO\_DATA(base, data)



can write to the PIO data register, turning the LED on and off. The PIO is just one of many SOPC peripherals that you can use in a system. To learn about the PIO core and other embedded peripheral cores, refer to Quartus II Version 10.1 Handbook Volume 5: Embedded Peripherals.

When developing your own designs, you can use the software functions and resources that are provided with the Nios II HAL. Refer to the Nios II Software Developer's Handbook for extensive documentation on developing your own Nios II processor-based software applications.

# 7.8 Debugging the Application

Before you can debug a project in the NIOS II IDE, you need to create a debug configuration that specifies how to run the software. To set up a debug configuration, perform the following steps:

1. In the **hello\_world.c** , double-click the front of the line where you would like to set breakpoint, as shown in **Figure 7-60**.

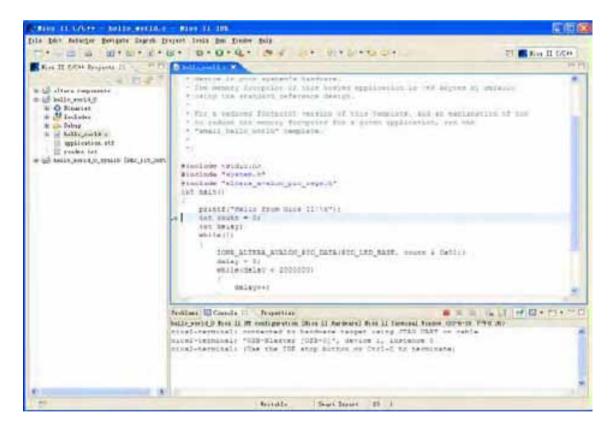


Figure 7-60 Set Breakpoint

- 2. To debug your application, right-click the application, **hello\_world\_0**, and select **Debug as** > **Nios II Hardware**.
- 3. If the **Confirm Perspective Switch** message box appears, click **Yes**.

After a moment, the main() function appears in the editor. A blue arrow next to the first line of code indicates that execution stopped at that line.

terasiC Terasic DE0-Nano User Manual

#### 5. Select **Run** > **Resume** to resume execution.

When debugging a project in the Nios II IDE, you can pause, stop or single step the program, set breakpoints, examine variables, and perform many other common debugging tasks.

*Note:* To return to the Nios II C/C++ project perspective from the debug perspective, click the two arrows >> in the top right corner of the GUI.

# 7.9 Configure System Library

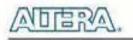
In this section you will learn how to configure some advanced options in the Nios II IDE. By performing the following steps, you can change all the available settings:

1. In the Nios II IDE, right-click **hello\_world\_0** and select **System Library Properties**. The **Properties for hello\_world\_0\_syslib** dialog box opens.

2. Click **System Library** in the tree on the left side. The **System Library** page contains settings related to how the program interacts with the underlying hardware. The settings have names that correspond to the targeted NIOS II hardware.

3. In the Linker Script box, observe which memory has been assigned for Program memory(.text), Read-only data memory(.rodata), Read/write data memory(.rwdata), Heap memory, and Stack memory, see Figure 7-61. These settings determine which memory is used to store the compiled executable program. You can also specify which interface you want to use for stdio, stdin, and stderr. You can also add and configure an RTOS for your application and configure build options to support C++, reduced device drivers, etc.

4. Select **onchip\_memory2** for all the memory options in the **Linker Script** box, as shown in **Figure 7-61.** 



TTO DESCRIPTION OF THE OWNER OF T	System Likeary				
Jada Pattare DDP Julij 2004 Julij 2004 Julij 2004 Julia Tajan 2004 Julian 2004 Julian 2004 Julian 2004 Julian 2004 Julian 2004 Julian 2004 Julian Patan Relation Rela	Target Backway	(reaction of the second of the	atalatan a	forder Teripi Caston Linker senipt Caston Linker senipt Commence Council Freque energy Council East oily decrementy Control Restricts data energy Control Restricts data energy Control Restricts data energy Control Restricts data energy Control	The second secon
	Diagram serve entits Plagmant.tre Capturenget Service Access MT Classic setty profiling Discovy Okingtowented conception builder Settingen Community	Ches and their infers balance server more balance server more balance off, or balance or the time shell declare	e++1	Die s opener stoptis tas Require the seary Release scoptic task tar (print)	

Figure 7-61 Configuring System Library Properties

5. Click **OK** to close the **Properties for hello\_world\_0\_syslib** dialog box and return to the IDE workbench.

*Note:* If you make changes to the system properties you must rebuild your project. To rebuild, right-click the hello\_world\_0 project in the Nios II C/C++ Projects tab and select Build Project.



# **Chapter 8**

**DE0-Nano Demonstrations** 

# **8.1 System Requirements**

Make sure Quartus II and NIOS II are installed on your PC.

# 8.2 Breathing LEDs

This demonstration shows how to use the FPGA to control the luminance of the LEDs by means of pulse-width modulation (PWM) scheme. The LEDs are divided into two groups, while one group dims the other group brightens, vice versa. Users can change the PWM wave's duty ratio and frequency to control the LED luminance and repetition rate.

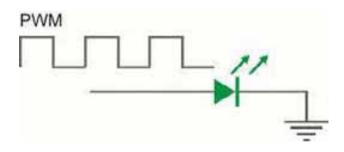
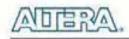


Figure 8-1 Shows a diagram of PWM signals to drive LED.



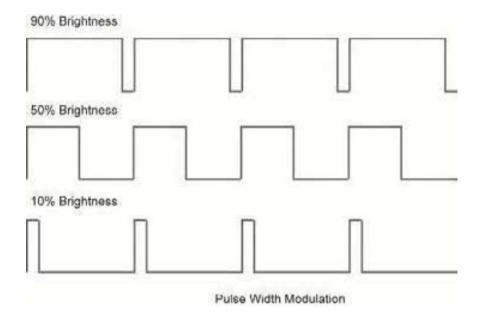


Figure 8-2 Pulse Width Modulation

Figure 8-2 shows the relationship between duty cycle and LED luminance.

# **Demonstration Source Code**

- Project directory: DE0\_NANO\_Default
- Bit stream used: DE0\_NANO.sof

# **Demonstration Batch File**

Demo Batch File Folder: DE0\_NANO\_Default\demo\_batch

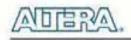
The demo batch file includes the following files:

• FPGA Configure File: DE0\_NANO.sof

# **Demonstration Setup**

- Make sure Quartus II and Nios II are installed on your PC.
- Connect USB cable to the DEO-Nano board and install the USB Blaster driver if necessary.
- Execute the demo batch file "DE0\_NANO.bat" under the batch file folder,

DE0\_NANO\_ Default Idemo\_batch. This will load the demo into the FPGA.

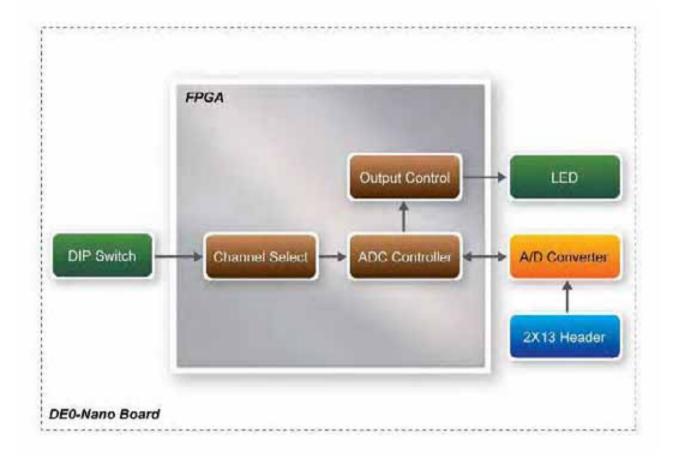


# 8.3 ADC Reading

This demonstration illustrates steps which can be used to evaluate the performance of the 8-channel 12-bit A/D Converter. The DC 3.3V on the 2x13 header is used to drive the analog signals and by using a trimmer potentiometer, the voltage can be adjusted within the range of  $0\sim3.3V$ . The 12-bit voltage measurements are indicated on the 8 LEDs. Since there are only 8 LEDs, only bit-4 through bit-11 from the ADC are represented on the LEDs.

# Design Concept

This section describes the design concepts for this demo. Figure 8-3 shows the block diagram.



#### Figure 8-3 ADC Reading Block Diagram

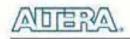
The ADC Controller reads the voltage from the A/D converter through a serial interface and displays its measurement on the LEDs. The on-board dip-switch determines which channel to read from. **Table 8-1** lists the DIP Switch settings and its corresponding ADC channel.



Table 5-1 DIP Switch Settings					
DIP Switch (SW1)	Setting	ADC Channel			
	0000	Analog_In0			
	0001	Analog_In1			
	0010	Analog_In2			
	0011	Analog_In3			
	0100	Analog_In4			
	0101	Analog_In5			
	0110	Analog_In6			
	0111	Analog_In7			

# Table 8-1DIP Switch Settings

**Figure 8-4** depicts the pin arrangement of the 2X13 header. Connect the trimmer to the ADC channel which is selected by the DIP Switches (Analog\_In0 ~ Analog\_In7).



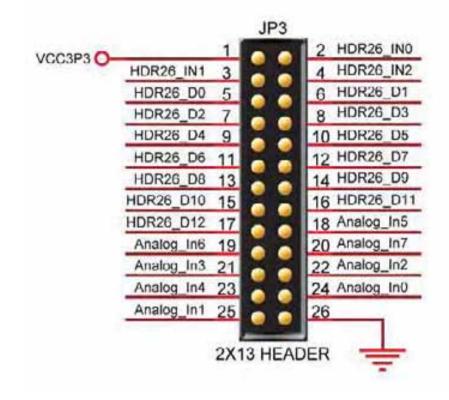


Figure 8-4 2X13 Header

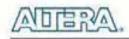
### **System Requirements**

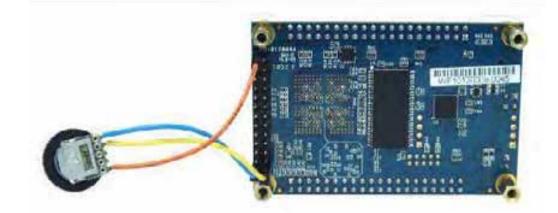
The following items are required for the ADC Reading demonstration

- DE0-Nano board x1
- Trimmer Potentiometer x1
- Wire Strip x3

### ■ Hardware Setup

• Figure 8-5 shows the hardware setup for the ADC Reading demonstration.





#### Figure 8-5 ADC Reading hardware setup

Note: the setup shown above is connected ADC channel 1.

# **Demonstration Source Code**

- Project directory: DE0\_NANO\_ADC
- Bit stream used: DE0\_NANO.sof

# **Demonstration Batch File**

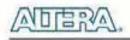
Demo Batch File Folder: DE0\_NANO\_ADC\demo\_batch

The demo batch file includes the following files:

- FPGA Configure File: DE0\_NANO.sof
- •

# **Demonstration Setup**

- Make sure Quartus II is installed on your PC.
- Connect the trimmer to corresponding ADC channel to read from, as well as the +3.3V and GND signals.
- Adjust the DIP switch according to the ADC channel connected
- Connect USB cable to the DEO-Nano board and install the USB Blaster driver if necessary.
- Execute the demo batch file "DE0\_NANO\_ADC.bat" under the batch file folder, *DE0\_NANO\_ADC\demo\_batch*. This will load the demo into the FPGA.
- Adjust the voltage using the trimmer and observe the measurements on the LEDs. Note a fully



lit LED bar indicates the voltage is 3.3V and similarly no LED lit indicates 0V.

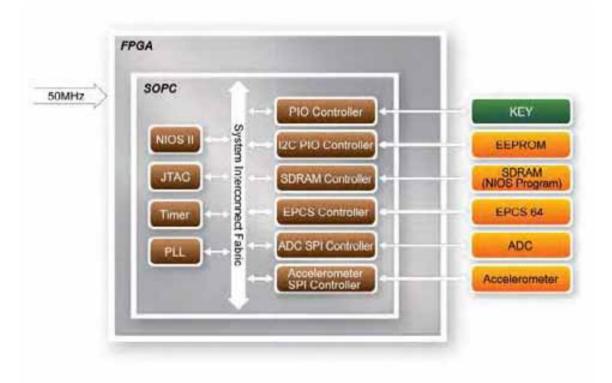
# 8.4 SOPC Demo

This demostration illustrates how to use the SOPC Builder to create a system with the following functions:

- Control accelerometer through 3-wire SPI interface
- Control analog to digital conversion through 4-wire SPI interface
- Access EEPROM memory through I2C interface
- Access EPCS memory

### System Block Diagram

This section describes the SOPC System Block Diagram of this demo, as shown in Figure 8-6.



#### Figure 8-6 SOPC Block Diagram

A 50 MHz Clock is required for the SOPC System. A NIOS II processor is included in the system for flow control. The PLL is used to generate clocks, including 100 MHz, 10 MHz and 2MHz. The NIOS II Processor and SDRAM are running at 100 MHZ. The SDRAM is used to store the NIOS II Program. The ADC SPI Controller is running at 2 MHz. The other peripheral controllers are running at 10 MHz. The ADC SPI Controller and the Accelerometer SPI Controller are custom SOPC component. The source code, for these two controllers, is located in the "ip" folder under this



Quartus II project. The other components are standard SOPC Builder components.

### ■ KEY

The KEY button is driven by PIO Controller with interrupt enabled. It is design to generate an interrupt event when users click KEY0 or KEY1. The interrupt event is used to terminate accelerometer and analog to digital conversion process in this demo.

For default, the interrupt is disabled in the PIO Controller. Users can enable it with the parameter setting as shown in below **Figure 8-7**.

PIO (F	Parallel VO)	Documentation
Basic Settings	and a second sec	
VADATA 1-32 bits:	2	
Direction.	CIBM	
	Onut	
	Onos	
	Oodené	
Output Port Report Vie	Let Sustaining the street of the	
Edge capture regi	iter	1
Synchronousty c	aphare .	-
Edge Type:	FALLING M	
	p for mign capiton register	
Trade Micherry		71
E Brudde Mildesen		
and the second		
Interrupt	EDGE W	)

Figure 8-7 PIO Controller

# Accelerometer Control

The accelerometer controller is a custom SOPC component developed by Terasic. The source code is available under the folder \DE0\_NANO\_SOPC\_DEMO\ip\TARASIC\_SPI\_3WIRE.

In this demo, the accelerometer is controlled through a 3-wire SPI. Before reading any data from the accelerometer, master should set 1 on the SPI bit in the Register  $0x31 - DATA_FORMAT$  register, as shown in below **Figure 8-8**, to set the device to 3-wire SPI mode.



Register 0x31—DATA_FORMAT (Read/Write)							
		105			132	101	
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justily	Range	

#### Figure 8-8 DATA\_FORMAT Register

The data format is configured as 10 bits, right-justify, +/- 2g mode. The output data rate is configured as 400 HZ. The X/Y/Z value is read using polling mode. Before reading X/Y/Z, the master needs to make sure data is ready by reading the register 0x30-INT\_SOURCE, as shown below **Figure 8-9**, and checking the DATA\_READY bit. In the demo, multiple-byte read of six bytes X/Y/Z, register from 0x32 to 0x37, is performed to prevent a change in data between reads of sequential register. Note, the output data is twos complement with DATAx0 as the least significant byte and DATAx1 as the most significant byte, where x represent X, Y, or Z.

Register 0x30—INT_SOURCE (Read Only)						
DATA READY	D6	D5	D4			
	SINGLE_TAP	DOUBLE_TAP	Activity			
D3	D2	D1	D0			
Inactivity	FREE_FALL	Watermark	Overrun			

#### Figure 8-9 Register 0x30

The SPI timing scheme follows clock polarity (CPOL)=1 and clock phase (CPHA)=1. (CPOL)=1 means the clock is high in idle. (CPHA)=1 means data is captured on clock's rising edge and data is propagated on a falling edge. The timing diagram of 3-wire SPI is shown below **Figure 8-10**:

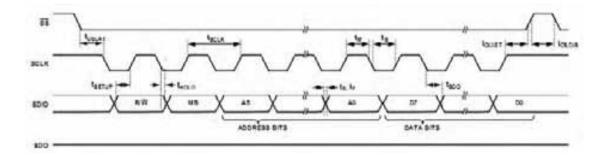
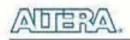


Figure 8-10 3-wire SPI Timing Diagram

### ADC Control

The Analog to Digital Conversion is controller through a 4-wire SPI interface with the timing dialog given below **Figure 8-11**. Note, the DIN signal is used to specify the channel (IN0~IN7) for the next data conversion. The DOUT signal is used to read the data conversion result whose channel is specified in previous transaction. The first conversion result after power-up will be on IN0. The output format of conversion result is straight binary.



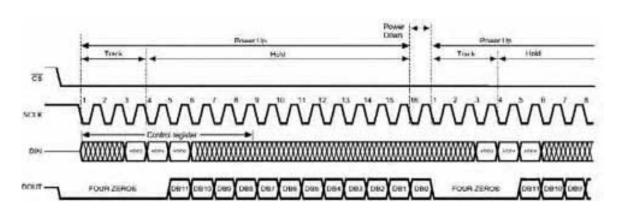


Figure 8-11 4-wire SPI Timing Diagram

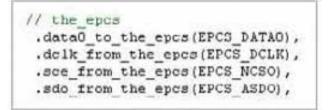
### **EEPROM Control**

EEPROM is accessed through the I2C interface. In this demo, I2C signal is toggle by NIOS II through the PIO controller. The I2C clock signal is driver by an OUTPUT PIO Controller and the I2C data signal is driver by a BIDIRECTION PIO Controller. The I2C C code is located in:

DE0\_NANO\_SOPC\_DEMO\software\DE0\_NANO\terasic\_lib\I2C.c

### **EPCS** Control

EPCS64 is accessed through the EPCS interface. In Quartus 10.0 or later, the EPCS pin assignment is required and should be connected the pins to EPCS Controller as shown below **Figure 8-12**:



#### Figure 8-12 EPCS interface connection

For the EPCS access functions, users can refer to:

 $DE0\_NANO\_SOPC\_DEMO\software\DE0\_NANO\terasic\_lib\Flash.c$ 



# **Demonstration Source Code**

- Project directory: DE0\_NANO\_SOPC\_DEMO
- Bit stream used: DE0\_NANO.sof
- NIOS II elf file: DE0\_NANO.elf

### **Demonstration Batch File**

• Demo Batch File Folder: DE0\_NANO\_SOPC\_DEMO\demo\_batch

The demo batch file includes the file:

- Batch File: test.bat and test\_bashrc
- FPGA Configure File: DE0\_NANO.sof
- Nios II Program: DE0\_NANO.elf

# **Demonstration Setup**

- Make sure Quartus II and Nios II are installed on your PC.
- Connect a USB cable to the DEO-Nano board and install USB Blaster driver if necessary.
- Execute the demo batch file "test.bat" under the batch file folder,
  - DE0\_NANO\_SOPC\_DEMO\demo\_batch. This will load the demo into the FPGA.
- After executing the batch file, a selection menu appears as follows:

us Nex 8 (05 10.1 [gec3]	- 🗆 🛪
Example designs can be found in /condrive/c/aligra/10.0/nipgZeds/gxamples	-
(You may add a startup script! cl/altera/18.0/mias2cds/user.bashrs) Usinu cable "USB-Blaster (USB-8)", device 1, instance Bx80 Resetting and pausing target processor: UK Initializing CPU cache (if present) of	
Downloaded 84KE in 1.4a (68.0KE/a) Devified OK	
Starting processor at address 8x820001C8	
nios2-terninal: connected to hardware target using JTAG WART on cable nios2-terninal: "USB Blaster (USB 0]", device 1, instance 0	
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)	
DE-Name Demo	
Selection Function: (B)ACCELEROMETER (L)ADG - (2)FEPROM - (J)AEPCY	
Select:	4

• Input "0" to start the accelerometer demo. The demo starts by displaying the accelerometer's chip ID, and then continues by displaying the X/Y/Z values every 1.0 second. To terminate the demo, press KEY0 or KEY1 on the DE0-Nano board. Upon exiting the demo, the selection menu will be displayed.





Miter II LDS 10.1 (gcc.)							D	×
Select:Domo ACCELEROMETER Id-E5h Hunitor Accorumeter Value, Fress XEY0 X=-20 ng, V=-4 ng, Z=872 ng X=-32 ng, Y=8 ng, Z=976 ng X=-20 ng, Y=8 ng, Z=956 ng X=-20 ng, Y=4 ng, Z=980 ng	or XE	¥1 tu	terninal	che	monitor	process.		
X-12 ny. Y-12 ny. Z-1001 ny X-36 ng. YA ng. Z-972 ny X23 ng. Y-W ng. Z-968 ny X28 ny. Y-8 ny. Z-980 ny								

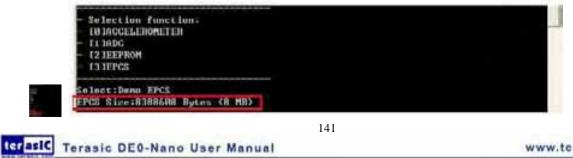
• Input "1" to start Analog to Digital Conversion demo. The demo repeatedly displays the voltage on eight channels. To terminate the process, press KEY0 or KEY1 on the DE0-Nano board. Upon exiting the demo, the selection menu will be displayed.

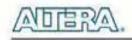
un Nies II FDS 10,1 [grc3]					- 0 ×
Select:Denu ADC Select:Denu ADC SHU-0.32 U SHU-0.32 U SHI-0.29 U SHI-0.37 U SHI-0.37 U SHI-0.37 U SHI-0.39 U	KEYS	ta teraina)	the monitor	jancess.	
986-0.23 V 207-0.32 V					4

• Input "2" to start EEPROM Content Dump demo. The demo displays the values in the first 16 bytes of the EEPROM. The demo automatically exists, and returns to the selection menu.

La Niss II FDS 10.1 [grc1]	- D ×
And a second	-
Select:Deno TETROH	
AddelW01 = FFA	
Adde (013 - ffh	
Addr (82) - ffh	
HAdw(W3) = FFK	
Addr(84) - FFh	
0ddr[05] - ffh	
Addression = ffb	
Addr[87] - ffh	
Add_1001 - FTh	
AddelN91 - FFh	
Addel107 - FFh	
Addr[11] - ffh	
Adde1121 - FFh	
Addel133 = FFh	
Addef141 - ffh	
Addr[15] - ffh	* ]

• Input "3" to start EPCS demo. The demo displays the memory size of EPCS. The demo automatically exists, and returns to the selection menu.





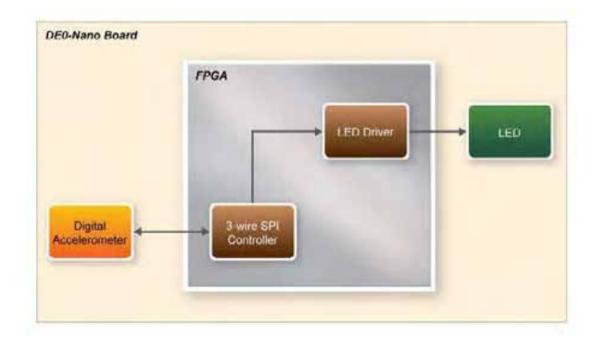
# 8.5 G-Sensor

This demonstration illustrates how to use the digital accelerometer on the DE0-Nano board to measure the static acceleration of gravity in tilt-sensing applications. As the board is tilted from left to right and right to left, the digital accelerometer detects the tilting movement and displays it on the LEDs.



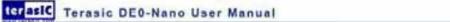
Figure 8-13 DE0-Nano on level surface

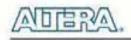
### Design Concept



This section describes the design concepts for this demo. Figure 8-14 shows the block diagram.

#### Figure 8-14 G-Sensor block diagram





In this demo, the accelerometer is controlled through a 3-wire SPI. Before reading any data from the accelerometer, the controller sets 1 on the SPI bit in the Register  $0x31 - DATA_FORMAT$  register. The 3-wire SPI Controller block reads the digital accelerometer X-axis value, to determine the tilt of the board. The LEDs are lit up as if they were a bubble, floating to the top of the board.

# **Demonstration Source Code**

- Project directory: DE0\_NANO\_GSensor
- Bit stream used: DE0\_NANO\_G\_Sensor.sof

# **Demonstration Batch File**

Demo Batch File Folder: DE0\_NANO\_GSensor\demo\_batch

The demo batch file includes the following files:

• FPGA Configure File: DE0\_NANO\_G\_Sensor.sof

# **Demonstration Setup**

- Make sure Quartus II is installed on your PC.
- Connect USB cable to the DEO-Nano board and install the USB Blaster driver if necessary.
- Execute the demo batch file "test.bat" under the batch file folder, *DE0\_NANO\_GSensor\demo\_batch*. This will load the demo into the FPGA.
- Tilt the DE0-Nano board from side to side and observe the result on the LEDs.

# 8.6 SDRAM Test by Nios II

Many applications use SDRAM to provide temporary storage. In this demonstration hardware and software designs are provided to illustrate how to perform memory access in QSYS. We describe how the Altera's SDRAM Controller IP is used to access a SDRAM, and how the Nios II processor is used to read and write the SDRAM for hardware verification. The SDRAM controller handles the complex aspects of using SDRAM by initializing the memory devices, managing SDRAM banks, and keeping the devices refreshed at appropriate intervals.

### System Block Diagram

Figure 8-15 shows the system block diagram of this demonstration. The system requires a 50 MHz



clock provided from the board. The SDRAM controller is configured as a 32MB controller. The working frequency of the SDRAM controller is 100MHz, and the Nios II program is running in the SDRAM.

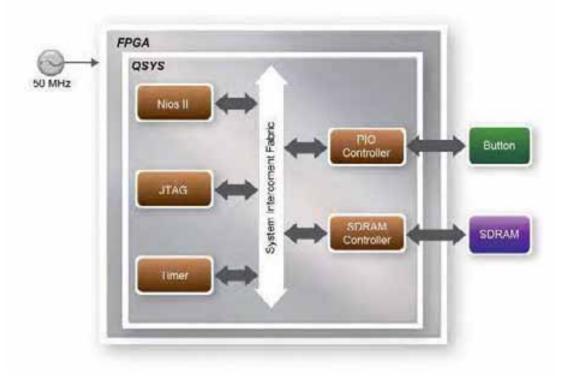


Figure 8-15 Block diagram of the SDRAM Basic Demonstration

The system flow is controlled by a Nios II program. First, the Nios II program writes test patterns into the SDRAM. Then, it calls Nios II system function, alt\_dcache\_flush\_all, to make sure all data has been written to SDRAM. Finally, it reads data from SDRAM for data verification. The program will show progress in JTAG-Terminal when writing/reading data to/from the SDRAM. When verification process is completed, the result is displayed in the JTAG-Terminal.

### Design Tools

- Quartus II 13.0 SP1
- Nios II Eclipse 13.0 SP1

### Demonstration Source Code

- Quartus Project directory: DE0\_NANO\_SDRAM\_Nios\_Test
- Nios II Eclipse: DE0\_NANO\_SDRAM\_Nios\_Test \Software

### Nios II Project Compilation

• Before you attempt to compile the reference design under Nios II Eclipse, make sure the project is cleaned first by clicking 'Clean' from the 'Project' menu of Nios II Eclipse.

### **Demonstration Batch File**

Demo Batch File Folder: DE0\_NANO\_SDRAM\_Nios\_Test \demo\_batch

The demo batch file includes following files:

- Batch File for USB-Blaster : DE0\_NANO\_SDRAM\_Nios\_Test.bat, DE0\_NANO\_SDRAM\_Nios\_Test.sh
- FPGA Configure File : DE0\_NANO\_SDRAM\_Nios\_Test.sof
- Nios II Program: DE0\_NANO\_SDRAM\_Nios\_Test.elf

### Demonstration Setup

- Make sure Quartus II and Nios II are installed on your PC.
- Connect a USB cable to the DE0-Nano board and install USB Blaster driver if necessary. Execute the demo batch file "DE0\_NANO\_SDRAM\_Nios\_Test .*bat*" under the batch file folder, DE0\_NANO\_SDRAM\_Nios\_Test \*demo\_batch*
- *After Nios II program* is downloaded and executed successfully, a prompt message will be *displayed in n*ios2-terminal.
- Press **KEY1~KEY0** of the DE0-Nano board to start SDRAM verify process. Press **KEY0** for continued test.
- The program will display progressing and result information, as shown in Figure 8-16.

```
Altera Nios II EDS 13.0sp1 [gcc4]
Pausing target processor: OK
                                                                                   ¥U
Initializing CPU cache (if present)
DK
Downloaded 61KB in 0.8s (76.2KB/s)
Uprified OK
Starting processor at address 0x00000104
nicc2-terminal; connected to hardware target using JTAG WART on cable
nios2-terminal: "US8-Blaster [US8-8]", device 1, instance 8
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)
SDRAM Test! Size:32MB (CPU Clock:10000000) :::::
Press any KEV to start test [KEVB for continued test]
 ====> SORAH Testing, Iteration: 1
aritm.
107. 207. 307. 407. 507. 607. 707. 807. 907. 1007.
 oad/verify
18% 28% 38% 48% 58% 88% 78% 88% 98% 188%
SDRAM test.Pass, 12 seconds
  ....> SDRAM Testing, Iteration: 2
rito.
 0% 20% 30% 40% 50% 60% 70% 80% 90% 100%
  ad/verify ....
```



Figure 8-16 Display Progress and Result Information for the SDRAM Demonstration

# **Chapter 9**

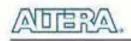


# **9.1 Programming the Serial Configuration Device**

This section describes how to program the serial configuration device with Serial Flash Loader (SFL) function via the JTAG interface. User can program serial configuration devices with a JTAG indirect configuration (.jic) file. To generate JIC programming files with the Quartus II software, users need to generate a user-specified SRAM object file (.sof) of the circuit they wish to put in the serial configuration device. Next, users need to convert the SOF to a JIC file. To convert a SOF to a JIC file in Quartus II software, follow these steps:

### • Convert SOF to JIC

- 1. Select File > Convert Programming Files...
- 2. In the **Convert Programming Files** dialog box, set the **Programming file type** field to **JTAG Indirect Configuration File** (.jic).
- 3. In the **Configuration device** field, specify the targeted serial configuration device, **EPCS64**.
- 4. In the **File name** field, browse to the target directory and specify an output file name.
- 5. Highlight the **SOF Data** row in the table, as shown in **Figure 9-1**.
- 6. Click Add File.
- 7. Select the SOF that you want to convert to a JIC file.
- 8. Click Open.



- 9. Highlight the Flash Loader and click Add Device, as shown in Figure 9-2.
- 10. Click **OK**. The Select Devices page displays.

ile <u>Window</u> ipecify the input files to fou can also import input future use. Conversion setup files					tup information cre	ated here for
Open Cor	ngersion Setup Data		-	Save	Conversion Setup.	
Output programming fil	c					
Programming file type:	JTAG Indirect Configu	ration File (.jic)				*
Options	Configuration device:	EPCS64	~	Mode:	Active Scrial	1V
File game:	output_file.jic					
Advanced	Remote/Local update	difference file:	NONE			•
Input files to convert		Dur	ou Hour		Start Address	Add Hex Data
File/Dat Flash Loader		1000-00-0	erties		Start Address	
SOF Data		Page 0			auto>	Add Sof Page
						Add Ele
						Remove
						Up
						Down
						Properties
				Genera	ate Close	Help

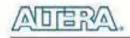
Figure 9-1 Convert Programming Files Dialog Box



Convert Progra	amming File -	DE/CD/DE_R	ano/DE	U_Nano	_ <b>▼1. 0. 9_CD</b> R	0 🔳 🗖 🔯
Specify the input files to a You can also import input future use.					etup information cre	ated here for
Open Cor	yersion Setup Data		_	Save	Conversion Setup	
Output programming file	0					
Programming file type:	JTAG Indirect Config.	ration File (.jic)				×
Options,	Configuration device:	EPCS64	~	Mode:	Active Scrial	1.4
File game:	output_file.jic					
Advanced	Remote/Local update	difference file;	YONE			<b>X</b>
Input files to convert						
File/Dat	a arta	Prope	rties	16	Start Address	Add Hex Data
Flash Loader		Page 0			auto>	Add Sof Page
DE0_NANO.sc	27 T	EP-4CE22F17				Add Device
						Hemove
						Up
						Down
						Properties
				Genera	ate Close	Help

#### Figure 9-2 Highlight Flash Loader

- 11. Select the targeted FPGA, Cyclone IV E EP4CE22, as shown in Figure 9-3.
- 12. Click OK. The Convert Programming Files page displays, should look like Figure 9-4.
- 13. Select the .sof file, and Click the **Properties**. Select Compression, click **OK**, as shown in **Figure 9-5**.
- 14. Click Generate.



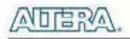
evice family	Device name	
APEX20K	EP4CE10	New
Arria GX Arria II GX	CP4CE115     EP4CE15	Import
Arria II GZ Cydone	EP4CE22	Export
Cyclone II Cyclone III	EP-1CE-10 DP-4CE55	Edit
Cyclone III LS Cyclone IV E	EP4CE6	Remove
Cyclone IV GX MAX II MAX V Stratix Stratix GX Stratix II Stratix II GX		Uncheck All
Stratix III Stratix IV		

Figure 9-3 Select Devices Page



Convert Progra	anning File - )	D=/CD/DE_	Nano/DE	O_Nano		) 🔳 🗖 🔀
ile <u>Window</u>						
Specify the input files to You can also import input tuture use. Conversion setup files	file information from of			version se		
Open Cor	rgersion Setup Data			Save	Conversion Setup.	
Output programming fil	e					
Programming file type:	JTAG Indirect Configu	ration File (.jic)				×
Options	Configuration device:	EPCS64	v	Mode:	Active Scrial	V
File name:	output_file.jic					
Advanced	Remote/Local update	difference file:	NONE			
Input files to convert						
File/Dat	a area	Prop	per lies		Start Address	Add Hex Data
EP4CC22		Page_0 <auto> EP4CE22E17</auto>		cauto>	Add Sof Page	
						Remove
						Lp
						Down
						Properties
				Genera	ate Close	

### Figure 9-4 Convert Programming Files Page



ie <u>Window</u>		_				
pecify the input files to ou can also import input uture use. Conversion setup files					tup information cre	ated here for
Open Cor	ngersion Setup Data		_	Save	Conversion Setup	
Output programming fil	c					
Programming file type:	JTAG Indirect Config.	ration File (.jic)				×
Options	Configuration device:	EPCS64	V	Mode:	Active Scrial	M
File game:	output_file.jic					
Advanced	Remote SOF F	ile Proper	ties			
Input files to convert	ОК	Canica		Help	t Address	
File/Dat Flash Loader EP4CE22 SOF Data DE0_NANO.se		Page_0 P4CE22E17	ur ves		auto>	Add Hex Data Add Sof Page Add Ele
						Remove
						Up
						Down
						Properties
				Genera	ate Close	Help

Figure 9-5 Compression the sof file

# ■ Write JIC File into Serial Configuration Device

To program the serial configuration device with the JIC file that you just created, add the file to the Quartus II Programmer window and follow the steps:

- 1. When the SOF-to-JIC file conversion is complete, add the JIC file to the Quartus II Programmer window:
  - i. Select **Tools > Programmer**. The **Chain1.cdf** window displays.
  - ii. Click Add File. From the Select Programming File page, browse to the JIC file.
  - iii. Click **Open**.



2. Program the serial configuration device by checking the corresponding **Program/Configure** box, a Factory default SFL image will be load (See Figure 9-6).

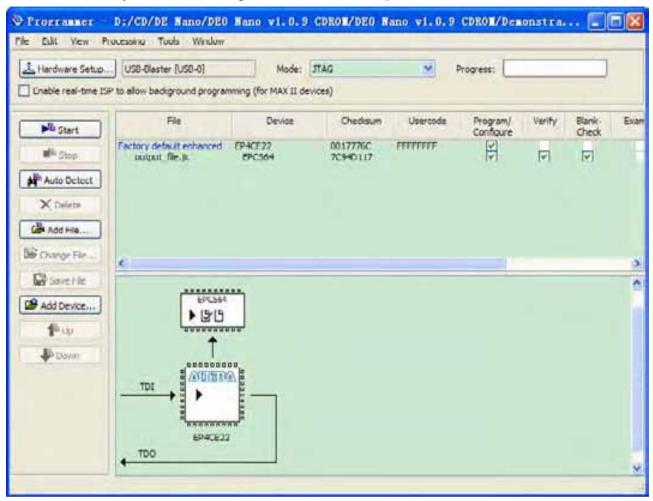


Figure 9-6 Quartus II programmer window with one JIC file

3. Click **Start** to program serial configuration device.

### **Erase the Serial Configuration Device**

To erase the existed file in the serial configuration device, follow the steps listed below:

- 1. Select **Tools > Programmer**. The **Chain1.cdf** window displays.
- 2. Click Add File. From the Select Programming File page, browse to a JIC file.
- 3. Click **Open.**
- 4. Erase the serial configuration device by checking the corresponding Erase box, a Factory



default SFL image will be load (See Figure 9-7).

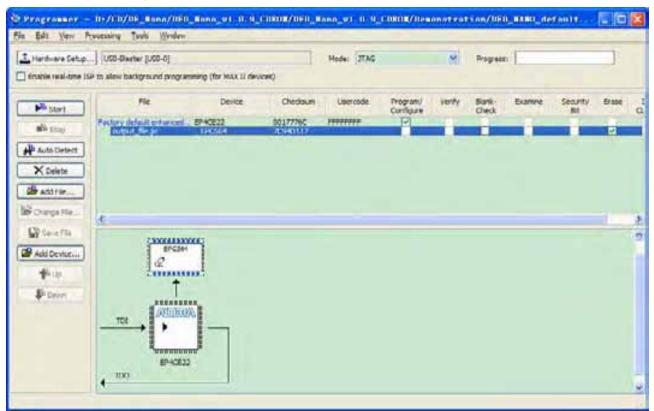


Figure 9-7 Erasing setting in Quartus II programmer window

5. Click **Start** to erase the serial configuration device.

# 9.2 EPCS Programming via nios-2-flash-programmer

Before programming the EPCS via nios-2-flash-programmer, users must add an EPCS patch file nios-flash-override.txt into the Nios II EDS folder. The patch file is available in the folder Demonstation\EPCS\_Patch of DE0-Nano System CD. Please copy this file to the folder [QuartusInstalledFolder]\nios2eds\bin (e.g. C:\altera\11.1\nios2eds\bin)

If the patch file is not included into the Nios II EDS folder, an error will occur as shown in **Figure 9-8**.

Using cable "USB Dlaster (USB 0)", device 1, instance ( Resettion and pausing target processor: OK	9×98
No EPCS layout data - looking for section [EPCS-010216.	1
Unable to use EPCS device	
Leaving target processor paused	

Figure 9-8 EPCS Message

# **9.3 Revision History**

Version	Change Log
V1.0	Initial Version (Preliminary)
V1.3	Add Table 3-1,3-2 and 3-3
V1.4	Modified Digital Accelerometer Description on page 31
V1.5	Modified ADC description on page 32
V1.6	Corrected Digital Accelerometer Schematic on page 23
V1.7	Modified Altera EPCS16 to be Spansion EPCS64
V1.8	Add SDRAM test section

# **9.4 Copyright Statement**

Copyright © 2012 Terasic Technologies. All rights reserved.

Always visit the DE0-Nano webpage for new applications.

We will continue providing interesting examples and labs on our DEO-Nano webpage. Please visit <u>www.altera.com</u> or <u>DEO-Nano.terasic.com</u> for more information.